Journal of Heuristics (2019) 25:753–792
https://doi.org/10.1007/s10732-018-9385-x

Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem

Haneen Algethami[1] · **Anna Martínez-Gavara**[2] · **Dario Landa-Silva**[3]

Received: 18 August 2017 / Revised: 21 March 2018 / Accepted: 16 July 2018 / Published online: 2 August 2018

Abstract

The workforce scheduling and routing problem refers to the assignment of personnel to visits, across various geographical locations. Solving this problem demands tackling numerous scheduling and routing constraints while aiming to minimise the operational cost. One of the main obstacles in designing a genetic algorithm for this problem is selecting the best set of operators that enable better GA performance. This paper presents a novel adaptive multiple crossover genetic algorithm to tackle the combined setting of scheduling and routing problems. A mix of problem-specific and traditional crossovers are evaluated by using an online learning process to measure the operator's effectiveness. Best operators are given high application rates and low rates are given to the worse. Application rates are dynamically adjusted according to the learning outcomes in a non-stationary environment. Experimental results show that the combined performances of all the operators were better than using only one operator used in isolation. This study provided an important opportunity to advance the understanding of the best method to use crossover operators for this highly-constrained optimisation problem effectively.

Keywords Genetic algorithms · Adaptive algorithms · Genetic operators · Routing · Scheduling · Workforce planning

1 Introduction

The workforce scheduling and routing problem (WSRP) is described as the assignment of personnel to visits, requested by customers, across different geographical locations.

✉ Haneen Algethami
hmgethami@tu.edu.sa

1 Collage of Computers and Information Technology, Taif University, Ta'if, Saudi Arabia

2 Estadstica y Investigacin Operativa, Universidad de Valencia, Valencia, Spain

3 ASAP Research Group, School of Computer Science, The University of Nottingham, Nottingham, UK

A type of WSRP arises in home health care where nurses and care workers should be assigned to visit patients in their homes to carry out some tasks, e.g. administering medication, monitoring serious illness and unstable health status and injections (Castillo-Salazar et al. 2015).

This problem combines scheduling and routing problems, both of which are known to be NP-Hard (Lassaigne and Rougemont 2012). The *scheduling aspect* of the problem assigns personnel to visits in order to fulfil work demands and other requirements. The *routing aspect* of the problem consists of generating routes for the workers to service customers across various locations within given time-windows. The objective is to minimise operational costs while attending the additional requirements expressed by customers, workers and the business.

Genetic Algorithms (GAs) have been shown to be effective approaches to find solutions for problems combining scheduling and routing where exact methods are less effective, e.g. Cowling et al. (2006), Mutingi and Mbohwa (2014) and Algethami et al. (2017).

A baseline GA was proposed by Algethami et al. (2016) that identified the best set of operators and parameters for each instance of WSRP. Despite its success in obtaining good solutions, the GA performance was limited by a computationally expensive parameter tuning method. Thus, an adaptive parameter control approach was proposed by Algethami and Landa-Silva (2017). In the later, the method maintained population diversity to enhance the performance of the GA. On the other hand, in the study by Algethami et al. (2016), there was no evidence that the performance of one crossover operator was superior to a group of operators during the run. Therefore, a random crossover exchange was also proposed by Algethami and Landa-Silva (2017), in addition to the parameter tuning method. Further improvements were suggested, especially on operators selection.

This study proposes an adaptive multiple crossover GA that uses a learning process to enhance the performance. The idea is to use an adaptive allocation rules on a mix of problem-specific and traditional crossovers, which are evaluated to measure the operator's effectiveness. Based on their efficiency, a roulette wheel concept is used to select an operator. The contribution of this study is to present a dynamic operator selection, as the search progress, as part of the multiple crossover framework, to reflect the crossovers behaviour in one iteration, according to the learning outcomes in a non-stationary environment.

This study claims that using a multiple crossover framework can enhance the GA efficiency when tackling WSRP scenarios. To the best of our knowledge, this adaptive multiple crossover GA has not been investigated for WSRP. To this end, the specific objectives of this study are:

- To present an adaptive multiple crossover GA that aims to enhance the baseline GA efficiency.
- To analyse the impact of crossover operators collaboration on the quality of solutions as well as algorithm speed.
- To carry out illustrative computational tests to show the utility of the proposed GA approach by comparing the performance of the various genetic operators consid-

ered. A total of 42 problem instances from six different real-world home health care scenarios were used in the experimental study.

In what follows, Sect. 2 reviews related work. Section 3 describes the WSRP, its formulation and the instances used in this paper. Section 4 outlines the proposed algorithm. Section 5 presents the experimental study and discusses the obtained results. The paper is then concluded in Sect. 6.

2 Related work

Recent research on the WSRP considered here is reviewed next. A mixed integer programming (MIP) with decomposition method (Laesanklang and Landa-Silva 2016) required considerable computation time (up to several hours) to solve larger problem instances with hundreds of tasks, indicating the need for faster solution methods. A Variable Neighbourhood Search (VNS) algorithm using problem-specific neighbourhood heuristics was presented in Pinheiro et al. (2016). The VNS obtained high-quality solutions and in less computation time for the same set of problem instances used in Laesanklang and Landa-Silva (2016). A number of studies have applied GAs to real-world problems where scheduling and routing are combined. Examples include Cowling et al. (2006) and Mutingi and Mbohwa (2014). In those works, the focus has been on algorithm design to obtain good solutions.

An investigation was presented by Algethami et al. (2017) comparing various genetic operators within two simple GAs, to tackle the subject problem. A more efficient GA was proposed by Algethami et al. (2016) with tuned parameters and using a customised solution representation to maintain feasibility of solutions. To enhance the GA efficiency, adaptive concepts were used by Algethami and Landa-Silva (2017) in addition to random crossover exchange.

Using a learning method, to adjust parameter values has proven to enhance the baseline GA performance (Algethami and Landa-Silva 2017). Nevertheless, selecting a random crossover, without any prior knowledge of its performance, was a straightforward procedure. It is still not clear which operator was the best for each problem set, during the run. Thus, this paper proposes a learning-based multiple crossover framework to tackle WSRP.

According to Maturana and Saubion (2008), multiple operators GAs has been used on real-world applications to benefit from the different performance of synchronised operators. Most researchers have utilised a group of operators, crossovers (Consoli and Yao 2014) or mutations (Contreras-Bolton et al. 2016), as part of the algorithm. For example, the study by Puljić and Manger (2013) that has investigated mixing eight crossover operators on routing problems. One crossover was selected at random, with all the operators having the same probability. This operator is then applied at the current iteration. The study suggested multiple crossovers method obtained better results than a traditional GA. Another study that proposed a multiple operator algorithm for routing problems was presented by Osaba et al. (2014). The aim was to test it for a variety of combinations of operators. As a result, the algorithm was proven to be efficient to solve routing problems, such as the Traveling Salesman Problem, the Capacitated

Vehicle Routing problem, Vehicle Routing with Backhauls and Multi-Depot Vehicle Routing Problem.

Two main operations are applied when using multiple operators, operators **selection** and **evaluation**. Adaptive operator selection (AOS) is identified as the online adjustments of the crossover function (Belluz et al. 2015), where all crossovers are used as one operator. However, each crossover has an application rate, that is relative to the crossover performance. The better the crossover, the higher the chances to be implemented more than a poor crossover. Thus, providing a variety of performances in one iteration. The diverse set of operators explore the search space differently, thus, more areas are explored.

Different AOS methods have been successfully used in the literature, including random operator exchange presented in Spears (1995), Puljić and Manger (2013), Osaba et al. (2013), Osaba et al. (2014) and Onieva et al. (2017). Adaptive operator allocation rules were also used as a learning-based operator selection function. Such as, probability matching (Tuson 1995), adaptive pursuit (Thierens 2005), multiple armed bandit (Auer et al. 2002), and sliding multiple armed bandit (Fialho et al. 2010; Li et al. 2014).

To measures a crossover effectiveness (performance), an operator evaluation process is used to analyse the impact of an operator application on the current search (Li et al. 2014). This method includes giving some reward to an operator according to the operator impact on the search. The next crossover is selected based on rewarded value. The best operator is scored higher in the credit registry and therefore this crossover is selected more.

In general, fitness improvement was used to measure a crossover performance, i.e. the better the new solutions, generated by an operator, the more chance of an operator to be applied. However, more performance measurements were explored and proven to provide good results, such as distance-based measurement (Consoli and Yao 2014) and an operator execution-time (Maturana and Saubion 2008).

Many considerations were taken into account when designing a multiple crossover GA, including the evaluation measurement used and the number of iterations required to obtain sufficient information on the operators' performance.

Existing research has focused on using a fixed number of iterations to determine "the best" operator. However, an AOS might converge to the best performing operator early in the search. Not to mention that different operators provide different results at different stages of the search. Thus, using preliminary information can have limited flexibility and leads up to a loss of performance. This observation was proved by Belluz et al. (2015), in which the use of a "dynamic" approach was established to be better than using a static one.

The intended contribution of this study is to present an efficient adaptive multiple crossover genetic algorithm (AMCAGA), as an adaptive variant of the GA presented in Algethami and Landa-Silva (2017).

The proposed GA uses allocation rules on a mix of problem-specific and traditional crossovers. Best operators are given high application rates, and low rates are given to the worse operators. Based on the learning outcomes, application rates are dynamically modified, to reflect the crossovers' behaviour at the current iteration. A roulette wheel concept is also used when selecting an operator, as an extension of the works by Tuson

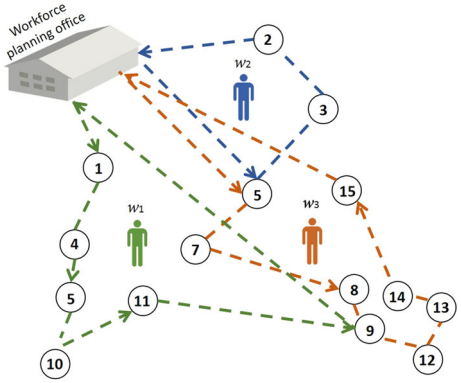

Fig. 1 A WSRP example, with 3 workers and 15 visit assignments (Colour figure online)

(1995) and Thierens (2005). However, in this study, the selection process is dynamic to provide a better reflection of the operator's performance, as the search progress. This study claims that using an active change of the application rates can enhance the GA efficiency when tackling WSRP scenarios. The objective is to show that the proposed method outperforms the GAs presented in Algethami et al. (2016) and Algethami and Landa-Silva (2017) in terms of results quality and run-time.

3 Problem description

A WSRP solution S is a daily plan of visits, i.e. a set of workers $W = \{w_1, w_2, \ldots, w_{|W|}\}$ assigned to perform a set of tasks $T = \{t_1, t_2, \ldots, t_{|V|}\}$ for customers at different locations, where V is the number of visit assignments. The assignment of a worker w to travel to a customer location in order to perform a task is called a visit.

Several features have been identified as important in solutions to WSRP scenarios, such as distance travelled and customers' and workers' requirements and preferences (Mankowska et al. 2014). Thus, a good quality solution should have low operational cost as well as all visits are assigned while satisfying the existing requirements.

For example, an illustration of a possible plan for a WSRP instance is presented in Fig. 1. In this example, 3 workers (w_1, w_2, w_3) are assigned to 15 different visits, located at different geographical areas. A path is plotted as a dotted line with a different colour for each worker, i.e., worker 1 is green, worker 2 is blue and worker 3 is orange. Worker w_1 is assigned to the set of visits (1, 4, 6, 10, 11, 9). Worker w_2 is assigned to the set of visits (5, 7, 8, 9, 12, 13, 14, 15). Worker w_3 is assigned to the set of visits (5, 2, 3). Note that two workers are assigned to visits 5 and 9, to perform different tasks for the same visit.

3.1 Problem constraints

Table 1 lists the objectives and constraints in the WSRP considered here as described in Laesanklang and Landa-Silva (2016). The binary decision variable $x_{i,j}^{w}$ indicates if

Table 1 Objectives and constraints in the WSRP

Objectives	Hard constraints	Soft constraints
Minimise the travelling cost	Assign all visits. Equation (2)	Respect workers area availability. Equation (13)
Minimise the payment cost	Respect visit time (no time-conflicts). Equation (9)	Respect workers time availability. Equations (10, 11)
Minimise penalty cost	Respect max working time per week. Equation (12)	Assign preferred workers to visits
	Assign qualified workforce. Equation (7)	Assign preferred workers with a specific skill
		Assign workers to preferred areas

a path connects two nodes (visit i and visit j) or not. The assignment $x^w_{i,j} = 1$ means that worker w travels from visit i to visit j, thus w makes both visits as given by constraint (1).

$$\sum_{i \in T} \sum_{j \in T} x^w_{i,j} = \sum_{n \in T} x^w_{j,n}, \forall w \in W \tag{1}$$

In WSRP scenarios, it is possible that some visits are left unassigned as there is not enough workforce or no worker has the required qualifications/skills. In such cases, an integer variable y_j is used to indicate the number of unsatisfied assignments for visit j, i.e. visit may require more than one worker (Laesanklang and Landa-Silva 2016; Rasmussen et al. 2012).

If visit j is fully assigned then $y_j = 0$, otherwise y_j takes a positive integer value equal to the number of workers required to the visit. Constraint (2) ensures this requirement is met even for visits that are unassigned, r_j is the number of workers required for visit j.

$$\sum_{w \in W} \sum_{i \in T} \sum_{j \in T} x^w_{i,j} + y_j = r_j \tag{2}$$

The path for each worker w should begin at a start location and terminate at an end location, e.g. their home or a central office. The start location and the end location of worker w are D_w and D'_w, respectively. The condition is enforced by constraints (3) and (4). Workers may leave their start location and enter their end location at most once (although the start and end locations may be different) as expressed by constraints (5) and (6), respectively.

$$\sum_{n \in T} x^w_{n,j} \geq \sum_{j \in T} x^w_{i,j}, \quad \forall w \in W, \quad \forall i \in T, \quad \exists n \in D \tag{3}$$

$$\sum_{n \in T} x^w_{i,n} \geq \sum_{i \in T} x^w_{i,j}, \quad \forall w \in W, \quad \forall j \in T, \quad \exists n \in D' \tag{4}$$

$$\sum_{j \in T} x^w_{i,j} \leq 1, \quad \forall w \in W, \quad \forall i \in D \tag{5}$$

$$\sum_{j \in T} x_{i,j}^{w} \leq 1, \quad \forall w \in W, \quad \forall j \in D' \tag{6}$$

Workers are required to have suitable skills for every assigned visit. Let q_j^w be a binary parameter that represent a qualification parameter, where $q_j^w = 1$ when a worker w has the skills to take visit j, and $q_j^w = 0$ otherwise. Only qualified workers can make the visit as indicated by constraint (7).

$$x_{i,j}^{w} \leq q_j^w, \quad \forall w \in W, \quad \forall i \in T, \quad \forall j \in T \tag{7}$$

Travelling between visit locations must be feasible in terms of travel time. Decision variable a_j^w takes a positive fractional value that gives the arrival time of worker w to the location of visit j. Note that the maximum arrival time value here is 1440 min, which is equivalent to the 24th hour of the day. Let a_i^w, a_j^w be the arrival times of worker w at the locations of visit i and visit j respectively. The arrival time at visit j must consider the time duration δ_i spent on performing visit i and the travelling time $t_{i,j}$ between visit i and visit j. This is enforced by constraint (8) where M is a large constant number.

$$a_j^w + M(1 - x_{i,j}^w) \geq a_i^w + x_{i,j}^w t_{i,j} + \delta_i, \quad \forall w \in W, \quad \forall i \in T, \quad \forall j \in T \tag{8}$$

A worker w must arrive at visit j within the given time-window. For visit j, the earliest arrival time is κ_j^L and the latest arrival time is κ_j^U. This requirement is enforced by constraint (9).

$$\kappa_j^L \leq a_j^w \leq \kappa_j^U, \quad \forall j \in T, \quad \forall w \in W \tag{9}$$

If a visit j assignment has time conflicts with the visit i assignment, $\tau_j^w = 1$. A time-conflict occurs when a worker is assigned to visits overlapping in time.

Time availability can be different for each worker according to their individual contracts. The time availability period for each worker is as follows. The shift start-time and shift-end time of the worker w are indicated by α_L^w and α_U^w respectively. However, in the scenarios tackled in here, visits can be assigned outside the workers shift but subject to a penalty cost. A binary decision variable $\omega_j^w = 1$ is introduced to indicate such penalisation. The time availability constraints for worker w are given by expressions (10) and (11).

$$\alpha_L^w - a_j^w \leq M(1 - x_{i,j}^w + \theta_j^w), \quad \forall w \in W, \quad \forall i \in D \cup T, \quad \forall j \in T \tag{10}$$

$$a_j^w + \delta_j - \alpha_U^w \leq M(1 - x_{i,j}^w + \theta_j^w), \quad \forall w \in W, \quad \forall i \in D \cup T, \quad \forall j \in T \tag{11}$$

Another working regulation tackled here is not to exceed the maximum working hours for each worker. Each visit j requires δ_j minutes to be completed. The maximum working hours for a worker w is given by h^w. Constraint (12) enforces this regulation.

$$\sum_{i \in V^S} \sum_{j \in T} x_{i,j}^w \delta_j \leq h^w, \quad \forall w \in W \tag{12}$$

Each worker is associated to a set of geographical regions defined by the service provider. A geographical region contains several visit locations and a visit location may have several visits to be assigned. Ideally, a worker should only be assigned to visits in those geographical regions. However, if necessary, a worker can be asked to travel to locations outside their geographical regions subject to some penalty cost. A binary parameter $\gamma_j^w = 1$ is defined to indicate that visit j is located in the workers regions and $\gamma_j^w = 0$ otherwise. A binary variable $\psi_j^w = 1$ to indicate that visit j assigned to worker w is outside the workers regions, and $\psi_j^w = 0$ otherwise. Constraint (13) presents the relation between these binary variables for the different possible cases.

$$\sum_{i \in T} x_{i,j}^w - \psi_j^w \leq \gamma_j^w, \quad \forall w \in W, \quad \forall j \in T \tag{13}$$

Some of the constraints expressed in the above MIP formulation are soft constraints in WSRP scenarios, including constraints (10) and (11) (workers may be asked to work outside their shift hours) and constraint (13) (workers may be asked to work outside their geographical regions). Later, preferences calculations are explained as they are used as part of the objective function.

3.2 Objective function

The objective function includes the *operational cost* and the *penalty cost*. The operational cost includes wages plus journey costs for all workers and is given by the accumulated cost $d_{i,j} + p_j^w$, where $d_{i,j}$ is the distance travelled between visit i to visit j and p_j^w is the cost of assigning worker w to visit j. These costs are set by the service provider in the HHC scenarios used here.

Since feasibility is not guaranteed in a WSRP solution, the penalty costs are tackled as the accumulated penalty for violations of the constraints presented in Table 1.

An assignment of worker w to visit j is made to a path connecting between two nodes and can be written as a tuple $\left(x_{i,j}^w, y_j, a_j^w, \psi_j^w, \theta_j^w, \tau_j^w\right)$. This assignment is composed of binary decision variables indicating violations occurrences on area availability (ψ_j^w), time availability (θ_j^w) and conflicting assignments (τ_j^w). If a visit j violates time availability, then $\theta_j^w = 1$. The same applies to area availability violation where $\psi_j^w = 1$, when violation occurs. Likewise, $\tau_j^w = 1$, if the assignment has time conflicts with the other assignment.

The non-satisfaction of preferences is also included in the penalty cost. There are three types of preferences including preferred worker-customer pairing, worker's preferred region and customer's preferred skills. There is a degree of satisfaction for these preferences when assigning a worker w to a visit j and is given by ρ_j^w which has a value that ranges between $[0, 3]$, where 0 means no penalty charged and 3 is full penalty.

For each assignment, the satisfaction value for each preference ranges between $[0, 1]$, from not satisfied to satisfied. There are four-level preferences: low (0.2), neutral (0.5), preferred (0.5), and most preferred (1.0). The satisfaction level is calculated by reverting it to a penalty, by subtracting it from the full satisfaction score, which is $3r_j$ for a visit j.

Table 2 Weights values for WSRP

Objective	Operational costs	Preferences penalty costs	Soft constraints penalty	Unassigned visits	Time-conflicts
Weight	λ_1	λ_2	λ_3	λ_4	λ_5
Value	$mileage/k$	1	$\frac{\lvert V\rvert}{2}$	${\lambda_3}^2$	${\lambda_3}^3$

The best solution should have: the least *operational cost* and the least *penalty cost*. A weighted sum is proposed to combine the objectives into a single scalar value (Pinheiro et al. 2016; Algethami et al. 2016). The objective function is written as in equation (14), where weights $\lambda_1, \ldots, \lambda_5$ are defined to establish priority between objectives (more about the weights used here next).

$$\begin{aligned} f(S) = \lambda_1 \sum_{w\in W}\sum_{i\in T}\sum_{j\in T}(d_{i,j} + p_j^w)x_{i,j}^w, \quad & \textit{operational cost} \\ + \lambda_2 \sum_{w\in W}\sum_{i\in T}\sum_{j\in T}(3r_j - \rho_j^w)x_{i,j}^w + \lambda_3 \sum_{w\in W}\sum_{j\in T}(\psi_j^w + \theta_j^w) & \\ + \lambda_4 \sum_{j\in T} y_j + \lambda_5 \sum_{w\in W}\sum_{j\in T}\tau_j^w, \quad & \textit{penalty cost} \end{aligned} \tag{14}$$

3.3 Weights calculations

The weights associated with each objective are set to values that reflect the difference between the priority levels, as suggested by Rasmussen et al. (2012) and Castillo-Salazar et al. (2016). The main goal is to maintain a balance of priorities for each instance based on each problem specifications. Thus, the weights calculations presented in Table 2 differ from one instance to another.[1]

Hard constraint violations are not allowed in the WSRP solution. However, time conflicts constraint is more difficult to satisfy. Thus, the highest priority level is given to minimise the conflicting assignments τ_j^w, where the associated weight λ_5 is set to the highest value. In this work, a solution with conflicting assignments is an infeasible solution.

The second highest priority to be minimised is the unassigned visits y_j where λ_4 is set to be very high, but still less than the value of λ_5. This is due to the service provider requirements for completing as many visits as possible. However, in this study, a chromosome representation ensures all visits to be assigned to workers, hence no unassigned visits violations. This is explained later in the next section.

In practice, the service provider may ask workers to undertake visits that are outside their time availability and/or geographical region. Thus, the next objective priority is λ_3, given to minimise the soft constraints penalty, i.e., the number of workers with time-availability violations ψ_j^w and the number of workers with area-availability violations

[1] The weights used here are available as "Evaluation Tool" (blue setting) at https://drive.google.com/drive/folders/0BzWzQ97MNBVZQXhicTUxRTNRWGc

θ_j^w. As presented by Table 2, the weight values for λ_3, λ_4 and λ_5 are relative to the number of the assigned visits, therefore only assigned visits are violated. On the other hand, if the service provider could not fulfil the highest preference level; the fourth priority is given to minimise the preferences penalty through λ_2.

Finally, the lowest priority is given to minimise the operational cost through λ_1. This weight value is presented in Table 2, where k is the operational cost. Hence, mileage is calculated for the assigned workers only.

Hard constraint violations are always penalised with a larger weight than soft constraints and therefore the total preference penalty cost. Accordingly, $\lambda_3 + \lambda_4 + \lambda_5 > 1$ while $\lambda_1 < 1$. Under those conditions, the objective is shifted to minimise the constraint violations, by moving the $f(S)$ cost-value closer to the feasible region.

3.4 Problem instances

Problem instances from six UK real-world HHC scenarios are used as instances of WSRP[2] in this study. There are 7 problem instances in each scenario for a total of 42 instances.

Table 3 shows the main features of each problem instance. Scenario A instances are considered the smallest, while instances in scenario F are the largest. Problem instances in scenario C are of disproportional nature as the number of workers is much larger than the number of visits.

There are some real-world problems that have similar characteristics to HHC, however, they are not tackled in here, as they are beyond the scope of the study. Such as security guard routing and rostering, worker allocation and maintenance personnel scheduling (Castillo-Salazar et al. 2016).

Figure 2 illustrates a group of box plots for each WSRP problem set, that present the overall total of the number of visits assignments $|V|$ and the total number of workers $|W|$. These plots provide a useful way to visualise the problem size by the vertical distance between the smallest value and the largest value, including any outliers.

As it can be seen from Fig. 2, the gap between the number of visits and the number of workers increases through the problem sets. Distributing assignments of visits over a set of workers is very significant in WSRP solutions, therefore any surplus between the number of visits assignments and the number of workers results in prioritising one over the other.

In some of the problem sets, the number of visits is larger than the number of workers available. Accordingly, any violations of visitations requirements constraints, such as conflicted visits, increases the value of the penalty function. On the other hand, in problem set C, the number of visits is less than the number of workers; hence, the opposite occurs with violations on workforce-related constraints.

The number of visits assignments and the number of available workforce determine the problem size of a WSRP. A small size WSRP in this study has around 50 tasks and 30 workers, as in the case of A and B. Whereas a mix size problem has around 80 visits and 800 workers or more, as in the case of C. A large problem size has more

[2] The instances data used here (A, B, C, D, E and F) are available as "Webroster set" at https://drive.google.com/drive/folders/0B2OtHr1VocuSMjAzd3FkWks2bDQ.

Table 3 Main features of the 42 home health care problem instances

	01	02	03	04	05	06	07	Mean	01	02	03	04	05	06	07	Mean
	A								B							
Number of visits	31	31	38	28	13	28	13	26	36	12	69	30	61	57	61	46
Number of workers	23	22	22	19	19	21	21	21	25	25	34	34	32	32	32	30
Number of areas	6	4	5	4	4	8	4	5	6	5	7	5	8	8	7	7
	C								D							
Number of visits	177	7	150	32	29	158	6	80	483	454	585	520	538	610	611	543
Number of workers	1037	618	1077	979	821	816	349	813	164	166	174	174	173	174	173	171
Number of areas	8	4	7	8	6	11	6	7	13	12	15	15	15	15	15	14
	E								F							
Number of visits	418	425	462	351	461	301	498	416	1211	1243	1479	1448	1599	1582	1726	1470
Number of workers	243	244	267	266	278	278	302	268	805	769	898	789	889	783	1011	901
Number of areas	13	14	15	13	15	13	16	14	45	46	54	47	59	44	64	51

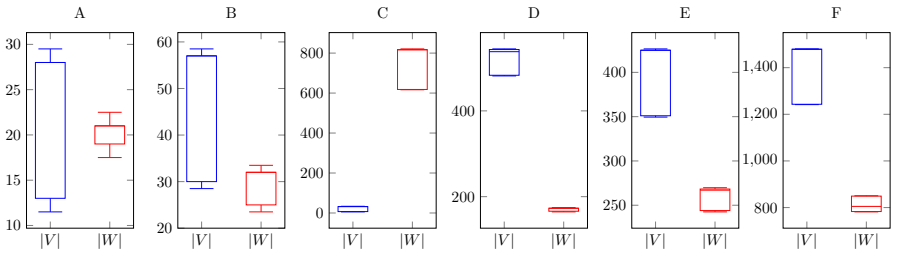

Fig. 2 Box plots comparisons of WSRP scenarios dimensions, shown as the number of visit assignments V and the number of workers W (Colour figure online)

Table 4 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances produced by the MIP solver in Laesanklang and Landa-Silva (2016)

Problem	$f(S)$	Cpt	Problem	$f(S)$	Cpt	Problem	$f(S)$	Cpt
A			**C**			**E**		
01	3.49	7	**01**	n/a	n/a	**01**	n/a	n/a
02	2.49	8	**02**	3.15	6	**02**	n/a	n/a
03	3	14	**03**	n/a	n/a	**03**	n/a	n/a
04	1.42	5	**04**	11.15	90	**04**	n/a	n/a
05	2.42	1	**05**	12.34	55	**05**	n/a	n/a
06	3.55	5	**06**	n/a	n/a	**06**	n/a	n/a
07	3.71	1	**07**	4.3	1	**07**	n/a	n/a
B			**D**			**F**		
01	1.7	21	**01**	n/a	n/a	**01**	n/a	n/a
02	1.75	2	**02**	n/a	n/a	**02**	n/a	n/a
03	1.72	6003	**03**	n/a	n/a	**03**	n/a	n/a
04	2.07	25	**04**	n/a	n/a	**04**	n/a	n/a
05	1.82	585	**05**	n/a	n/a	**05**	n/a	n/a
06	1.62	184	**06**	n/a	n/a	**06**	n/a	n/a
07	1.79	300	**07**	n/a	n/a	**07**	n/a	n/a

than 400 visits and around 270 workers, as the case of D and E. Problem set F has an average of 1470 visits and 900 workers, which characterise it as even larger than large instances.

Table 4 presents an overview of optimal solutions provided by a mixed integer programming (MIP) solver in Laesanklang and Landa-Silva (2016). The cost-values, i.e. objective function values, is denoted as $f(S)$. The computational-times is denoted as Cpt.

Researchers have reported that real-world instances of the WSRP are difficult to solve (Misir et al. 2011; Castillo-Salazar et al. 2016). This was proven further by the MIP, where it was unable to provide solutions for 24 instances denoted as *n/a*.

The problem size affects the algorithm performance, where small instances are less challenging computationally. Hence, MIP has provided optimal solutions for A, B and

some of the C problem set. Nevertheless, the MIP ran out of memory for D, E and F problem sets. Thus, larger problems require more efficient algorithms with problem specific-information.

4 Adaptive multiple-crossover genetic algorithm (AMCAGA)

This section describes the proposed adaptive aspects of the AMCAGA as an extension of the GA approach in Algethami et al. (2016) and the diversity-based adaptive GA (AGA) presented in Algethami and Landa-Silva (2017).

Algorithm 1 shows the steps of the proposed adaptive multiple crossovers genetic algorithm (AMCAGA). The proposal here is a combination of the adaptive genetic algorithm (AGA), which includes a control mechanism for genetic operator rates P_c and P_m, and the adaptive multiple-crossover genetic algorithm (AMCGA). The reason for selecting AGA as part of AMCAGA is because it has provided the best results in comparison to other adaptive variations implemented for WSRPs, as stated in Algethami and Landa-Silva (2017).

Algorithm 1 Adaptive Multiple-Crossover Genetic Algorithm (AMCAGA)

Require: A crossover rate P_c, a mutation rate P_m, set of crossovers $\chi = x_1, x_2 \ldots x_L$, *WSL* for all visits.
1: Create a population P of M individuals using *WSL*
2: **repeat**
3: Evaluate each individual in P with equation (14)
4: **for** $9M/20$ times **do**
5: Select $p_1, p_2 \leftarrow P$
6: **end for**
7: Select $x_i \leftarrow \chi$ with P_{x_i}
8: $(o_1, o_2) \leftarrow x_i(p_1, p_2)$ with P_c, for all pairs of parents
9: $Score_i \leftarrow$ Performance of x_i
10: Update $P_{x_i} \leftarrow Score_i / Score_{Sum}$
11: $(o'_1, o'_2) \leftarrow FCF(o_1, o_2)$ with P_m, for all pairs of offspring
12: $(o'_1, o'_2) \rightarrow P'$, for all pairs of offspring
13: P best $M/10$ individuals $\rightarrow P'$
14: **if** P has stagnated for a number of generations **then**
15: **if** $P_c > 0.45$ AND $P_c < 1.0$ **then**
16: Update P_c according to equation (15)
17: **else**
18: Reset P_c
19: **end if**
20: **if** $P_m > 0.1$ AND $P_m < 0.60$ **then**
21: Update P_m according to equation (16)
22: **else**
23: Reset P_m
24: **end if**
25: **end if**
26: $P' \rightarrow P$
27: **until** termination condition is met

The proposed AMCAGA works as follows. First, an initial population P of M individuals (1-day plans) is created based on an indirect chromosome encoding to ensure solutions feasibility (line 1). At the start of each generation, $9M/20$ pairs of parent individuals are selected using binary tournaments (line 4). With some probability P_{x_i}, a crossover x_i is selected, using the roulette wheel concept (line 5). This crossover, with some probability P_c, is applied at a time to each pair of parents to generate two offspring (line 6). The operators' performances are evaluated and recorded as *scores*(lines 7–8). With some probability P_m, each offspring goes through a flat-cost flip (FCF)

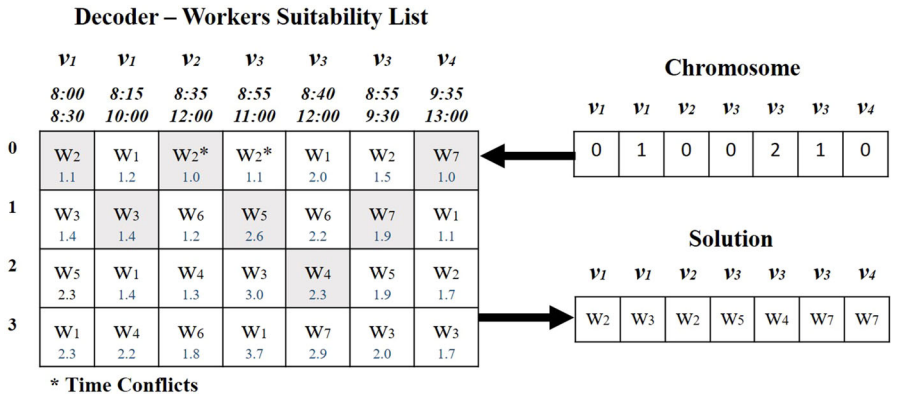

Fig. 3 An indirect chromosome encoding scheme illustration

mutation operator (line 9). At the end of each generation t, the population is updated using an elitism strategy, where the $M/10$ best individuals from the current population are kept. Along with the $9M/20$ offspring individuals generated, the new population of M solutions is formed. If there are no cost value improvements on the best so far solutions for a number of generations, the GA is considered to have stagnated. Thus, P_c and P_m are modified at every generation, depending on the results obtained in the previous one, to maintain a diverse population and therefore improve the effectiveness of the search (lines 11–22). The indirect chromosome encoding, genetic operators, adaptive parameter updates and multiple crossover mechanism are described next.

4.1 Indirect chromosome encoding

According to Osaba et al. (2014), the crossover process is not efficient for the optimization capacity of the technique when it is applied to routing problems using path encoding. Thus, indirect chromosome encoding scheme was proposed in our earlier study (Algethami et al. 2016). The aim is to construct feasible solutions by having suitable workers only. To do so, a *worker suitability list (WSL)* is created for each visit k, , where $k = 1, 2 \ldots, |V|$. Suitable workers are ranked by a penalty score $M_{k,j}$, i.e., the lower the score value, the better suited is a worker. The score is calculated for a worker w_j by estimating the impact of assigning that worker to that visit, considering incurred operational cost and penalty cost due to preferences and availability restrictions.

Initial population is created randomly by generating a vector of $|V|$ integers between 0 and $L_k - 1$, where L_k is the length of WSL for a visit k. When a worker is being considered for a visit, the solution is evaluated for time-conflicts. If it occurs, by the random assignments, the next worker in the WSL for that visit is considered until a suitable worker is found with no time-conflicts arising.

Figure 3 illustrates an example of the indirect chromosome encoding for a day plan with seven visits. Each visit has a WSL of four suitable workers, with the best worker for that visit at the top, followed by the next best worker and so on. Below the

chromosome, the decoded solution shows the actual worker assigned to each visit. On the right, the encoded solution is shown with an index of the workers as in the WSL for each visit. Time-conflicts are indicated with *.

For example, w_2 is assigned to both v_1 and v_3 while w_3 is assigned to v_2. No time-conflict arises because v_1 and v_3 do not overlap. However, w_2 is assigned to v_4 and a time-conflict arise as v_3 overlaps with v_4. Then, the next most suitable worker that does not provoke a time-conflict, in this case w_5, is assigned to v_4. The WSL decoder in this indirect chromosome encoding scheme helps to assign suitable workers to visits while avoiding time-conflicts. The penalty scores are shown in Fig. 3 are not used during the generation of initial solutions, they are utilized for the tailored genetic operators described in Algethami et al. (2016).

Note that the representation is designed to include all assigned visits $|V|$, thus chromosome length varies according to the problem-size.

4.2 Genetic operators

The AMCAGA incorporates various genetic operators including some problem-specific ones that utilise heuristics to generate improved offspring (Algethami et al. 2016). All operators used are taken from the study (Algethami et al. 2016). This is because they were proven to perform well under WSRP. They are applied as part of the AMCAGA, with probability P_c. For an effective collaboration, a mix of traditional and problem-specific operators are used. Four well-known operators are considered, including one-point crossover (1PX) (Hartmann 1998), two-point crossover (2PX) (Hartmann 1998), uniform crossover (UX) (Hartmann 1998) and half uniform crossover (HX) (Hartmann 1998). Three problem-specific operators are considered, specially designed for the solution representation considered here. Including flat-costs crossover (FCX) (Algethami et al. 2016), partially-matched flat crossover (PMFCX) and flat-cost flip mutation (FCF) (Algethami et al. 2016).

The reason for using a diverse set of operators is to allow dominance of one crossover over the others. The problem focused operators, on the other hand, have the advantage of using heuristics on WSRP instances, where they perform better than traditional crossovers as stated in Algethami et al. (2016). These operators extend the WSL capabilities. The operators used in this study are as follows.

1. *1PX* A point is selected at random between 1 and the chromosome length. The genes before this point are copied from one parent and the genes after are copied from the other parent.
2. *2PX* Two points are selected at random, between one and the chromosome length. Alternating segments are swapped to get new off-springs.
3. *UX* The number of crossing points are not fixed. Instead, a mixing ratio (50%) is used to choose a uniform random real number u from interval $\langle 0, 1\rangle$. If $u \leq 0.5$; then genes are swapped from parent's individuals, to be copied into offspring. Otherwise, copy the same genes with their current positions from the parent's individuals (Chu and Beasley 1998).

4. *HX* Similar to UX, a mixing ratio is used. However, exactly half of the non-matching gens are swapped. Thus, the Hamming distance (number of differing gens between the two parents) is calculated and divided by two.
5. *FCX* Uses penalty scores that were initially calculated in the WSL at the start of the GA. These values are denoted as 'flat-cost', where each worker w_j has a penalty score according to their suitability to work in visit i. FCX goes through each of the V positions in the parent's chromosome. A gene-wise comparison is enforced, for each gene in i^{th} position, with respect to the WSL estimated penalty scores M_{i,p^i}. The best suitable worker is given to offspring (o_1) and the other worker for offspring (o_2).
6. *PMFCX* This crossover selects a segment within two cutting points. Positions of genes are reversed between these points and the FCX is applied to fill the rest of the offspring.
7. *FCF* Introduces new workers to the chromosome, even if the worker are not suitable for the corresponding visit. A random position i of the chromosome is replaced. Hence, FCF increases the diversity by the random process, even if the best worker for visit i is not selected. $p_i = (0, 1, 0, 0, 2, 1, 0)$. If positions 3 is selected at random, gene 0 is replaced by a random number within the list of visit i in WSL, to generate the child chromosome $o_j = (0, 1, 3, 0, 2, 1, 0)$.

4.3 Adaptive parameter rates

Initial parameter values, P_c and P_m, were selected after an offline tuning method (Algethami et al. 2016). After the new population is created, if there are no cost value improvements on the best so far solutions for a number of generations, the GA is considered to have stagnated. This means that the search process could be trapped in a local optimum, or that the population could be concentrated in the same region of the solution space. Thus, P_c and P_m are modified at every generation, depending on the results obtained in the previous one, to maintain a diverse population and therefore improve the effectiveness of the search. Such feedback from the search is used to generate the next population. To avoid early convergence, a diversity-based scale is used to calculate the required change in the adaptive features. Population diversity in here is measured from two aspects.

- *The genotype space* (Morrison and De Jong 2002), denoted as $Diversity_g$. It is the distribution of pairwise differences between individuals in a population.
- *The phenotype space*, denoted as $Diversity_p$. It is the population variance, i.e., how far each individual in the population is from the mean value.

The two measurements accommodate different views of the loss of diversity. Combining these methods can overcome those difficulties provided by one measurement used in isolation (Črepinšek et al. 2013).

Population diversity has an effect on the setting of parameter values (Algethami and Landa-Silva 2017). For example, if instances are relatively small, a large population size M is required as a result of the large degree of similarity between individuals in P. When similar individuals are recombined, inbreeding occurs, and no additional

diversity is added through crossover. For a large instance, there is less chance to have similar solutions, thus, a small M is sufficient.

Whenever the best solution found by the technique has not been improved in the last generation, rates updated, according to the calculations presented by Algethami and Landa-Silva (2017). This means that the search process did not evolve correctly and that it is necessary to diversify the population through adaptive parameter rates.

Equation (15) calculates the updates on P_c value, where $Q_1 = 45\%$ and $Q_2 = 100\%$. Equation (16) calculates the updates on P_m value, where $Q_1 = 10\%$.

$$P_c = \left[\frac{Diversity_g}{Diversity_p} * (Q_2 - Q_1)\right] + Q_1 \tag{15}$$

$$P_m = \frac{\frac{Diversity_p - Diversity_g}{Diversity_p} \times Q_1 + \frac{f(S)_{max} - f(S)}{f(S)_{max} - f(S)_{min}} \times Q_1}{2} \tag{16}$$

A resetting process is also used for the adaptive rates P_c and P_m to forgets the rates history if the rates values are out of range, where rates starts from $P_c = 45\%$ and $P_m = 0.10\%$. This forgetting step means ignoring all the previous feedback process that led up to the inflation of the adapted values. More details of the AGA framework is described in Algethami and Landa-Silva (2017).

4.4 Adaptive multiple crossover framework

In relation to the multi-crossover feature, the proposed AMCAGA uses more than one crossover operator, from a set of crossovers $\chi = x_1, x_2 \ldots x_L$ of a size L, which are alternated during the execution, similar to Osaba et al. (2014). However, the replacement strategy used in this paper differ to the one described in Osaba et al. (2014), by using allocation rules rather than random replacement.

At the beginning, a number of the crossover functions are applied, until the next generation is created. This number is indicated as the memory size σ. One operator is assigned at random and it is randomly replaced by another crossover . All crossovers are applied uniformly, to ensure all crossovers are used, while allowing repetitions. There is at least $\frac{1}{L}$ chance for each operator to be selected.

The operators' performances are evaluated and recorded as *scores* for a number of iterations during the learning process (also denoted as a cycle of a size υ). For each crossover, the accumulated scores are stored into the **crossover reward matrix (CRM)**, considered as a reward registry for that crossover in the current cycle. Scores are then transformed into **application rates** P_{x_i}, *where* P_{x_i} is the probability of applying a crossover x_i. The operators' performances are evaluated and recorded for each cycle, where application rates are dynamically adjusted as the search progress.

Better operators have a higher score value, and therefore a higher probability to be utilised more, and week operators are not left without a chance. For example, given a set χ of six crossovers operators, with application rates as follows: $x_1 = 23\%$, $x_2 = 4\%$, $x_3 = 11\%$, $x_4 = 34\%$, $x_5 = 17\%$ and $x_6 = 12\%$. Crossover x_4 has the highest application rate at this cycle, henceforth x_4 has more chance to be implemented. At some point of the search, when x_4 performance changes for the worse, its application

rate decreases. Application rates of other crossovers also vary accordingly, causing a shift in the search.

Each operator is implemented independently; however, all operators share the same cycle until it resets. Hence, AMC is considered as a mix between natural systems (recombination operators) and synthetic systems (autonomous agents). The *natural system* aspect is reflected by the collective performances of the crossovers, onto a population (Eiben et al. 2007). The *synthetic systems* aspect is reflected by sharing a cycle as part of the learning process that directs their application (Panait and Luke 2005). Details for crossovers performance measurements and CRM construction is explained next.

4.4.1 Performance measurements

Scores are considered as performance indicators that enforce the rewarding mechanism, *where* $\forall x_i \in \chi$, a score is given at each iteration, where $i = 1, 2 \ldots, L$. To calculate the scores, performance measurements are used to model the behaviour of different operators in the current stage of the search.

Different types of measurements are used in the literature, such as fitness-based measurement (Whitacre et al. 2006; Li et al. 2014), distance-based measurement (Consoli and Yao 2014) and combined measurement and operator execution-time measurement (Maturana and Saubion 2008). In this study, the later can be relative to the problem size for WSRPs. Hence, the time is not absolute in real-world settings, and therefore this measurement is excluded.

Parameter settings are affected by the population diversity (Algethami and Landa-Silva 2017). Therefore, distance-based measurement is considered here along with the fitness-based measurement. Performance measurements used in this study are as follows.

1. **Fitness-Based**, This method is selected to maximise the cumulative improvement, as a historical fitness record of the cost value of an offspring (o) and its parent (p), where $f(o) < f(p)$. When there is an improvement on the overall cost value, the score increases by one. This value ensures the convergence by selecting crossovers with better offspring quality, that eventually improves the overall performance of a GA.
2. **Distance-Based**, This method is selected to ensure a level of diversity among generated solutions, by calculating the distribution of the pair-wise differences between an offspring and its parent. If the percentage is 60% or more, the score increases with respect to the *hamming distance* H between two individuals. This value is selected to prevent inbreeding among generations. The value of H is calculated as the number of assignments in which the corresponding genes are different. Figure 4 illustrates an example of the dissimilarities percentage calculations (presented in white cells) between a parent and an offspring.
3. **Hybrid Approach**, This method is selected to harness the power of the two above measurements. When there is an improvement in the fitness or there exist differences between parents and offspring, the score increases. This means that with every crossover evaluation, the score value increases gradually. If only one measurement has proved an operator efficiency, the score value increases by one. On the

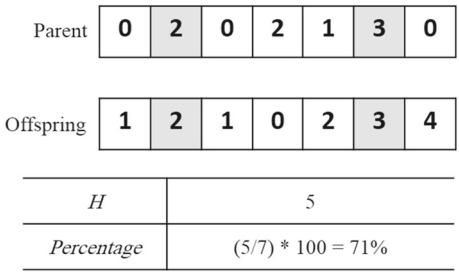

Fig. 4 An illustration of the dissimilarities percentage (presented in white cells) between two individuals (parent, offspring)

other hand, if both performance measurements have proved an operator efficiency simultaneously, the score value increases by two.

Each crossover is evaluated in isolation, after each application, with a performance measurement. The purpose of such separate evaluation is to obtain their relative score, under the evaluation criterion. In the event of one crossover is selected repeatedly, the score value increases gradually, based on the feedback from the online learning mechanism in AMC. Details for application rates calculations is described next.

4.4.2 Application rates

At each iteration (cycle), one of three above mentioned performance measurements (fitness, distance or hybrid) evaluate all crossovers, and then gives a score value $\forall x_i \in \chi$, where $i = 1, 2 \ldots, L$. For the next iteration, scores are updated, until all cycles are completed. The accumulated score value over different cycles is then transformed to an application rate $P_{x_i} = \frac{Score_i}{Score_{Sum}}$, where $Score_{Sum} = \sum_{i=1}^{L} Score_i$.

When recording the accumulated scores, a $2 \times L$ matrix is used, noted as crossover rewards matrix (CRM), i.e., the score value in $(1, i)$ is added to the value in $(2, i)$. The $2 \times L$ CRM is used to allow faster computation, rather than using a L size matrix.

As a result of scores updates, a gradual change in the P_{x_i} value exists in CRM, with no rapid increase or decrease in the overall application rates. This process is repeated, for each cycle. Note that the application rates are always modified before the next cycle (iteration).

Figure 5 shows an example of the CRM construction process, for three crossovers x_1 x_2, and x_3, with a cycle size $\upsilon = 3$. For every cycle, two steps are executed. First, scores are retrieved for each crossover. In this case, the first cycle has a $Score_1 = 3$, $Score_2 = 4$ and a $Score_3 = 5$. Next, the scores are shifted to $(2, i)$. Once a score is shifted to position $(2, i)$, a value of 1 is stored at position $(1, i)$, to ensure the application of all crossover functions, regardless of a crossover performance. If a crossover application has resulted in no improvement or even worse solution, it is still applied in the AMC. The accumulated scores are then calculated by adding up the scores at the position $(1, i)$ to the scores at the position $(2, i)$. The active calculation of the application rates maintains a balance between the most effective operators, which provide good results, and week crossovers, which provide poor results.

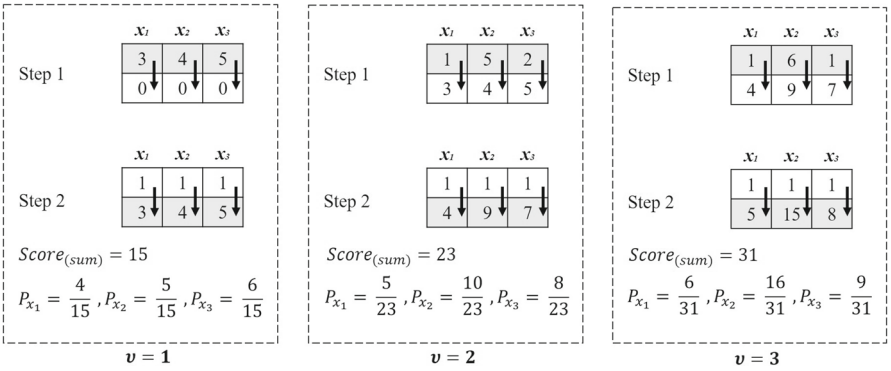

Fig. 5 An example illustration of the application rates calculations

Table 5 Parameter settings for AMCGAs and AMCAGA

Parameter	A	B	C	D	E	F	Fixed	Initial Values
Mutation rate P_m	50%	50%	30%	10%	10%	10%	AMCGAs	AMCAGAs
Crossover rate P_c	50%	50%	50%	100%	100%	100%	AMCGAs	AMCAGAs
Population size M	500	500	500	250	250	250	AMCGA, AMCAGA	–
Memory size σ	250	250	250	125	125	125	AMCGA, AMCAGA	–

5 Experimental study and results

The implementation has focused on applying the AMCGA on the GA presented on Algethami and Landa-Silva (2017) to capture the advantages of this method on the GA performance. For these set of experiments, the main elements used were the indirect chromosome representation and the FCF mutation. In this paper, six operators were used to allow dominance of one crossover over another, including 1PX, 2PX, UX, HX, FCX and PMFCX.

Since parameter settings and running times stated in Algethami et al. (2016) were successful in providing good results, each run in these experiments was executed with the same set of settings as shown in Table 5. Note that the crossover rate P_c and the mutation rate P_m values stated in Table 5 are used as initial settings for the AMCAGAs. However, these values are fixed when used with AMCGAs.

Similar to genetic operators rates, each WSRP problem set has different memory size σ. This value indicates the number of crossovers required to obtain enough information for the learning process, where $P_c \times \sigma = const$. Thus, the crossover rate P_c affect the value of σ, to record the crossover performances on at least $M/2$ of the population.

For example, in smaller instances, $P_c = 50\%$, therefore, less crossovers are applied. Hence, larger σ is required to obtain sufficient information about the crossovers performances. In larger instances, $P_c = 100\%$, this means that more crossovers are applied, and a large memory size is not necessary.

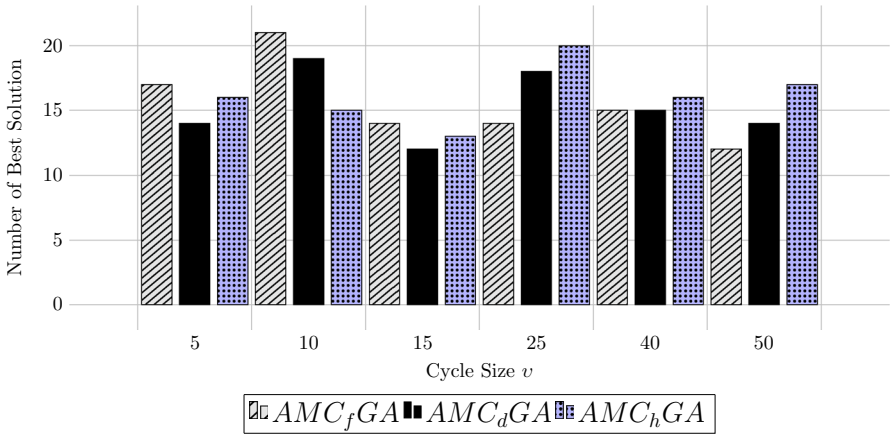

Fig. 6 Total number of best solutions produced by different rewarding mechanisms (AMC_fGA, AMC_dGA and AMC_hGA) (Colour figure online)

Different cycle settings were used here, where $\upsilon = \{5, 10, 15, 25, 40, 50\}$. The reason for selecting these values was due to the preliminary experimentation. The testing revealed that a cycle of a size less than 5, had a slow learning process. Accordingly, all operators were applied equally. A cycle of a size larger than 50 had a low accuracy of the learning factor, in which crucial information was lost. For example, a crossover operator might be the best performer at the start of a cycle, yet it can get worse in the same cycle.

Results were grouped by the performance measurement method applied. Each measurement is noted as follows: fitness improvements (AMC_fGA), dissimilarities between individuals (AMC_dGA) and the hybrid approach (AMC_hGA). The best performing method was selected to be included in AMCAGA and combined with adaptive parameter control method AGA, presented in Algethami and Landa-Silva (2017).

Each algorithm was executed 8 times (runs). This means that for each of the 42 problems instances, there were 8 × 6 (cycles) × 3 (methods) = 144 runs, seeded with the same initial population.

5.1 Performance of AMCGAs with different cycle sizes

This set of experiments aims to investigate the effectiveness of the feedback mechanism in the AMCGAs, when using different cycle sizes. To do so, different cycle sizes were examined with respect to the method applied.

Figure 6 shows the total number of the best solutions for all problem instances. Each bar at the x-axis represents a rewarding method applied with a cycle of size υ. Methods shown in the plot are: AMC_fGA (grey bars with striped lines), AMC_dGA (black solid bars) and AMC_hGA (blue bars with dots). The higher the bar the better, i.e. the more "best" values.

It is clear that all methods provided relatively similar values with a slight increase on one of the methods over the others. Still, AMC_fGA obtained the highest number

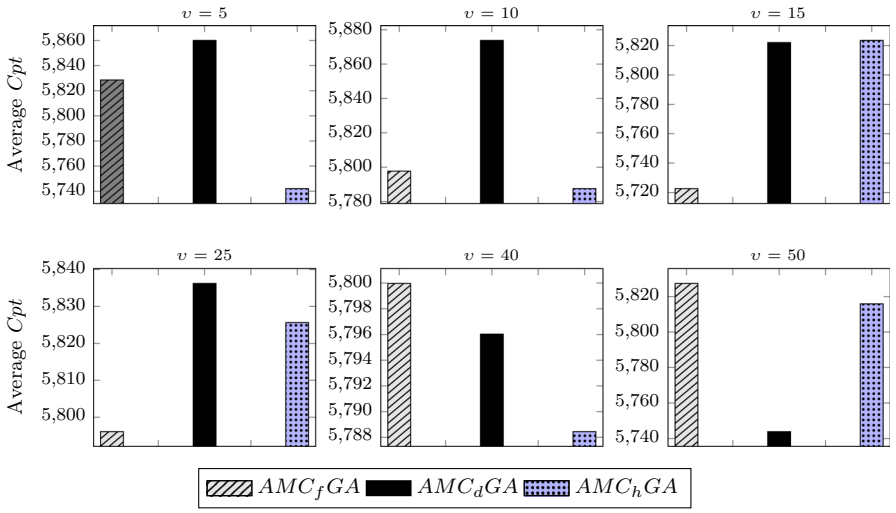

Fig. 7 Average computation-time (in seconds) produced by AMC variations with different cycle sizes υ (Colour figure online)

of best solutions. For AMC_fGA and AMC_dGA, the highest number of best solutions was the when $\upsilon = 10$. For AMC_hGA, on the other hand, the highest number of best solutions was when $\upsilon = 25$.

This result is related to scores given to an operator throughout the search, in which the accumulated score increases or decreases gradually. An operator performance at the start of the cycle might changes drastically by the end of the cycle, especially when a large υ value is used. Nevertheless, $\upsilon > 25$ was proven to be less effective for all methods. Interestingly, the highest number of best solutions was obtained while using AMC_fGA, with $\upsilon = 10$. Hence, there is no need for larger υ value to keep long historical records.

With one performance measurement, i.e., AMC_fGA and AMC_dGA, small changes in the score values occur. This provided enough time for updating the application rate, for each crossover, and therefore these crossovers were used in the current population. On the other hand, a large υ was required when the hybrid performance measurement was applied. This is because of a large and discrete change in the score values that occurs as the outcome of combining both performance measurements. Nevertheless, if the υ was small; there was not enough time to reward all operators. Thus, one operator will dominate the algorithm, resulting in an uneven distribution of the application rates. Henceforth, a larger cycle-size was required.

In summary, the larger the υ, the vast increase in the score value. This process has more effect on the search than having scored in small cycles, in which are ignored earlier in the search. Thus, υ value can affect a crossover dominance in the AMCGA method over another.

Figure 7 illustrates the overall average computational times in seconds, on problem sets A to F, for all methods. Each sub-figure corresponds to a cycle size υ and presents the average computational time used by AMC_fGA (grey bars with striped lines),

AMC_dGA (black solid bars) and AMC_hGA (blue bars with dots). The lower the bar the better, i.e. the less computational time.

On average, it was apparent from the plots that different methods provide solutions in approximately similar computational times.

For AMC_fGA, it was more computationally expensive to make large adjustments in the score values, especially when smaller υ was used. On the contrary, more time existed when using larger υ values, providing a shift in the score values when AMC_fGA was used. Thus, the largest computational times for AMC_fGA, among all methods, was recorded when $\upsilon = 40$ or 50.

For AMC_dGA, more time was spent on calculating the scores, that resulted in more computational time, especially when υ was equal 5 to 25. This was the outcome of the diversity-based calculations, computed gene by gene, and executed V times for each individual to calculate the scores accurately.

Fitness-based calculations were faster than the diversity-based calculations. However, when υ was equal 40 to 50, the sensitivity of the distance-based has provided good solutions in less computational time. This is because that distance-based measurement method was more sensitive to the change of performances than fitness-based measurement. Note similar execution time to AMC_dGA with AMC_hGA was recorded, with $\upsilon = 15$.

Contradictory to AMC_dGA, AMC_hGA was not affected by the convergence speed of the algorithm. The combinations between the diversity-based measurement and the fitness-based measurement in AMC_hGA have obtained better computational times than the separate methods. This is because AMC_hGA have averaged the performances of both measurements by allowing the use of the calculations within the time limitations. The reason for this was the large increase in the score values due to both calculations. As a result, the scores were computed in less time while using AMC_hGA.

The next set of experiments is designed to use one cycle-size per method, selected according to the highest total number of best solutions (see Fig. 6). Thus, $\upsilon = 10$ was chosen for AMC_fGA as well as for AMC_dGA, and $\upsilon = 25$ was selected for AMC_hGA.

5.2 Overall comparison between AMCGAs methods

The second set of experiments was designed to identify if there is a significant difference between the proposed AMCGAs variations or not. The work by García et al. (2008) has recommended the use of Friedman analysis as a non-parametric statistical test to establish statistical significance in EAs. Thus, a two-way Friedman analysis was used to measure the significant difference between groups of data when the dependent variable being measured was ordinal.

In this study, an IBM SPSS 22 two-way analysis was used to compare the variances of seven related-samples, with a significant level of $\alpha = 0.05$ and 95% as a confidence interval. Table 6 provides the results generated by the Friedman analysis on the 42 problem instances including the mean value, the standard deviation, the minimum cost-value, the maximum cost-value and the mean rank. The results

Table 6 Non-parametric Friedman's Statistical results combined with performances metrics

Method	Mean	SD	Min	Max	Mean rank	Dev %	# Best	Score
AMC_fGA	287.07	699.78	1.18	3495.55	1.99	0.13%	0.50	0.65
AMC_dGA	**286.74**	**699.11**	1.18	3496.73	**1.82**	**0.05%**	**0.64**	**0.73**
AMC_hGA	287.23	700.10	1.18	3496.36	2.19	0.16%	0.33	0.46

Bold text refers to the best result

presented in the mean rank column show the methods order based on the statistical analysis, where a low rank indicates the best method while a high rank indicates the worse method ranked overall. All problem instances were used to set the sample size of one method as large as possible, this increases the probability of accepting or rejecting the null hypothesis. Three additional values were calculated and used to measure the performance of each algorithm **Dev.** is the average percentage deviation from the best-known value (best solution of all the algorithms applied), **Best** is the fraction of instances in a set for which an algorithm matches the best-known value (best solution of all the algorithms applied). Additionally, the **Score** is the fraction of the instances for which a competing algorithm 'wins', i.e. produces better solutions than the configuration being scored. This score is calculated as $((q \times (p-1)) - r)/(q \times (p-1))$, where p is the number of methods compared, q is the number of problem instances, and r is the number of instances in which the $p-1$ competing configurations find a better result. Hence, the best score value is 1, when $r = 0$, and the worst score value is 0, when $r = q \times (p-1)$. The best results are highlighted in bold.

We applied the Friedmann non-parametric statistical to the data in Table 6 and obtained a p value of $0.02153 < 0.05$, degrees of freedom = 2 and $\chi^2 = 7.677$. This indicates the existence of significant performance differences among the three methods.

To examine where the differences actually occurs, an additional analysis was implemented. A post-hoc statistical analysis is needed in all cases. Holm's test was chosen to detect the significance difference among all variations. The Holm procedure is an example of a step-down procedure. Step-up procedures start testing hypothesis H_m and step up through the sequence while retaining the hypotheses. The procedure stops at the first rejection (for example H_i), and $H_1, \ldots, H_i$ are all rejected.

In this case, Holm's method obtains the p-values higher than the significance level, that is to be interpreted in the sense that we do not have enough evidence to reject the null hypothesis.

However, the descriptive statistics and the measures explained above revealed that AMC_dGA method had the lowest mean value, standard deviation value, mean rank value and Dev% value. This method had also the highest fraction of the number of best solutions and the score values. This finding suggests that different methods applied to different datasets can obtain different results and henceforth each problem set benefited from each method accordingly.

AMC_dGA obtained the best values in the above table. Thus, it can be argued that all variations were suitable for WSRP while AMC_dGA was slightly better than all

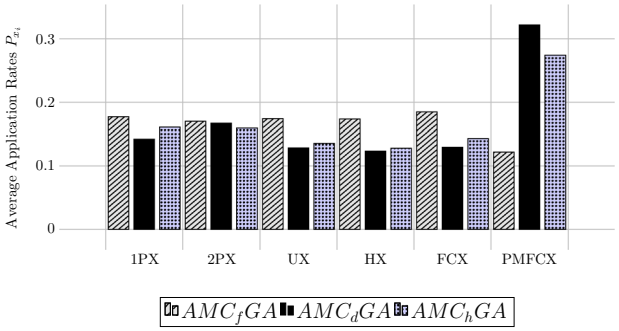

Fig. 8 Average application rates values used under different rewarding mechanisms (AMC_fGA, AMC_dGA, and AMC_hGA) (Colour figure online)

the proposed multiple crossovers algorithms. Using the distance-based measurement has provided more accurate scores adjustments in comparison to other methods.

5.3 Crossovers dominance based on application rates distribution

This section explores the change in the crossovers' application rates (P_{x_i}) to investigate the effectiveness of one crossover over the other, in each AMCGA. Thus, this set of experiments aims to identify the most used operator in each method applied, with respect to the performance measurement.

Each bar at the x axis in Fig. 8 illustrates the average application rate P_{x_i} values, for all problem sets. All three methods were applied, with the set of crossovers, i.e. 1PX, 2PX, UX, HX, FCX and PMFCX. The AMC_fGA is plotted in grey bars with striped lines, the AMC_dGA is plotted as black solid bars and the AMC_hGA is plotted as blue bars with dots.

Among all crossovers applied with AMC_fGA, the values for the average P_{x_i} were relatively similar with a slight increase on FCX. This result further proves the observations stated in Algethami et al. (2016), that identified the FCX as one of the best crossovers for WSRP problem sets. Thus, this operator has been utilised more when a fitness-based performance measurement was utilised.

Nevertheless, the highest average application rate P_{x_i} among all crossovers was obtained by PMFCX, while using AMC_dGA. Hence, PMFCX was utilised more often when distance-based performance measurement was applied.

Faster calculations were required when using AMC_dGA and AMC_hGA. Thus, FCX was used less under these methods. On the other hand, the PMFCX crossover was utilised more, due to its fast calculations as a result of mixing heuristic approach with the traditional PMX crossover method.

Traditional crossovers have provided poor quality solutions that were comparatively similar to their parents. Still, using them alongside the problem-specific method can ensure various performances in the GA.

Figure 9 illustrates a group of box plots comparisons to show overall patterns of response of change in application rates (P_{x_i}), for each crossover operator. Crossovers

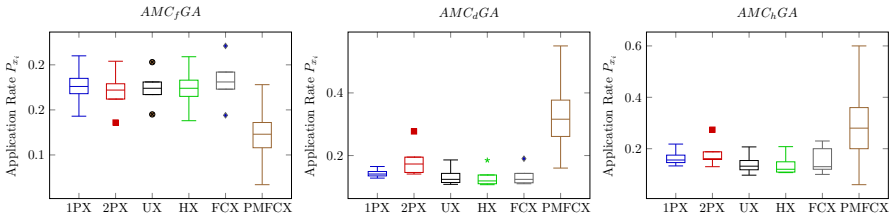

Fig. 9 Box plots comparisons of the average application rates range shown for crossovers 1PX (blue), 2PX (red), UX (black), HX (green), FCX (grey) and PMFCX (brown) (Colour figure online)

distribution corresponds to the current environment state if there is change in the P_{x_i} values, i.e the bigger the box plot the more diverse is the P_{x_i} values to the correspondent crossover.

Each sub-figure corresponds to the method applied, AMC_fGA, AMC_dGA and AMC_hGA. Each box-plot illustrates the average application rates (P_{x_i}) range, for each crossover and are shown in 1PX (blue), 2PX (red), UX (black), HX (green), FCX (grey) a PMFCX (brown).

As it can be seen from the plots, PMFCX box plot was lower than all other plots in AMC_fGA, i.e. less change in P_{x_i} values. The opposite occurs for AMC_dGA and AMC_hGA, where P_{x_i} values were more diverse. This indicates that PMFCX was updated more frequently. This result further supports the previous finding discussed in this section.

5.4 Effect of using a rewarding process on AMCGA

The third set of experiments aims to compare AMCGA methods with existing GA that was tailored for WSRP. The detailed results shown in Table 7 compares the indirect GA described in Algethami et al. (2016) (noted as GA) with the AMCGA variations (AMC_fGA, AMC_dGA and AMC_hGA). Crossovers used with indirect GA were specified for each problem set as follows. for A and B GCX, for C PMGreedyX, for D, E and F FCX. The FCF mutation was also used.

In addition to the indirect GA, the AMCGAs were compared against a variation of AMCGA that used a uniform choice of operators, noted as AMC_rGA. The gal for this comparison is to investigate the performance of AMCGAs even when excluding the reinforcement learning process. For each problem instance, the table shows the solution quality, noted as $f(S)$, and the computational time in seconds, noted as Cpt, in which the best solution was found.

Best values are highlighted in bold. If more than one method achieved the same result, among cost-best equals, the time-best is highlighted in bold.

As it can be seen from Table 7, the proposed methods AMC_fGA, AMC_dGA and AMC_hGA outperform the GA and AMC_rGA in terms of computational time, in particular AMC_fGA and AMC_dGA. This indicates that the rewarding process in the adaptive methods were more efficient time-wise. The percentage of best cost-values overall solutions are as follows. The GA 14.29%, AMC_rGA 30.95%, AMC_fGA

Table 7 Results of the cost values $f(S)$ and computational time Cpt (in seconds) produced by the proposed methods GA, AMC_rGA, AMC_fGA, AMC_dGA and AMC_hGA for 42 WSRP instances

	GA		AMC_rGA		AMC_fGA		AMC_dGA		AMC_hGA	
	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt
A										
01	3.49	34.08	3.49	3.90	3.49	2.27	3.49	1.85	**3.49**	**1.65**
02	2.96	396.77	2.49	5.07	2.54	2.34	2.49	2.62	**2.49**	**1.42**
03	3.65	436.92	3.01	153.54	**3.02**	**14.83**	3.02	58.57	3.02	115.48
04	1.42	229.01	1.42	41.76	**1.42**	**41.54**	1.42	67.69	1.42	66.38
05	2.42	0.98	2.42	1.07	2.42	0.81	2.42	1.26	**2.42**	**0.42**
06	3.61	159.83	3.55	4.64	**3.55**	**3.21**	3.55	8.30	3.55	8.93
07	3.71	0.83	3.71	1.02	3.71	0.81	3.71	1.42	**3.71**	**0.32**
B										
01	1.76	422.62	1.72	146.60	**1.72**	**141.90**	1.72	276.01	1.72	262.21
02	1.75	78.57	1.75	0.64	1.75	0.71	1.75	1.22	**1.75**	**0.34**
03	1.90	629.30	1.79	329.10	1.78	555.93	1.78	570.47	**1.78**	**435.03**
04	2.08	29.51	2.08	4.77	**2.08**	**4.06**	2.08	21.25	2.08	7.28
05	1.96	434.63	1.91	417.83	**1.90**	**490.40**	1.91	483.60	1.92	558.09
06	1.71	343.74	1.64	105.39	1.65	80.54	**1.64**	**72.75**	1.65	137.81
07	1.87	502.96	1.80	248.04	**1.79**	**57.24**	1.80	244.68	1.79	144.08
C										
01	115.13	846.87	115.01	597.34	114.81	513.89	**114.75**	**566.20**	114.88	641.68
02	3.15	0.46	3.15	0.98	3.15	0.67	3.15	0.86	**3.15**	**0.20**
03	104.83	657.11	105.02	659.60	**104.65**	**675.85**	104.89	434.64	104.82	535.76
04	11.15	2.41	11.15	2.65	11.15	1.84	**11.15**	**1.79**	11.15	2.25

Table 7 continued

	GA		AMC_rGA		AMC_fGA		AMC_dGA		AMC_hGA	
	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt
05	12.34	3.38	12.34	16.09	12.34	2.94	12.34	68.90	**12.34**	**2.58**
06	180.98	591.22	180.98	367.50	**180.80**	**301.95**	180.80	431.25	180.86	377.21
07	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**
D										
01	173.15	2989.34	171.39	3140.40	171.08	3408.46	**170.71**	**3334.63**	171.08	3255.77
02	169.56	3502.67	167.50	3448.17	168.11	3046.96	**167.31**	**3405.58**	168.31	3450.06
03	184.49	3502.43	180.93	3211.17	180.86	3361.38	**180.86**	**3102.37**	181.55	3390.17
04	171.83	3484.81	168.68	3338.98	168.42	3323.28	168.37	3476.31	**167.86**	**3412.84**
05	162.32	3589.68	161.93	3382.11	161.99	3266.39	**161.86**	**3276.84**	161.92	3113.17
06	178.80	3446.80	178.16	3268.95	178.35	3209.79	**178.16**	**3353.26**	178.28	3278.63
07	178.85	3423.81	178.65	3263.43	178.71	3367.56	178.65	3347.14	**178.59**	**3271.47**
E										
01	3.75	3314.33	1.18	3420.64	1.18	3366.72	1.18	3404.99	**1.18**	**3206.71**
02	3.86	3570.60	1.21	3394.69	**1.21**	**3311.93**	1.21	3399.96	1.21	3431.21
03	5.18	3371.76	1.22	3345.45	**1.21**	**3428.17**	1.22	3460.71	1.22	3395.24
04	3.27	3584.67	1.30	3419.12	1.29	3512.18	**1.29**	**3493.56**	1.30	3432.41
05	5.15	3585.41	**2.24**	**3287.96**	2.24	3469.36	2.25	3424.19	2.25	3450.09
06	3.56	3559.72	1.30	3406.16	**1.30**	**3233.54**	1.30	3356.92	1.30	3377.66
07	3.92	3574.02	**1.73**	**3339.36**	1.73	3486.11	1.73	3452.58	1.73	3375.31

Table 7 continued

	GA		AMC_rGA		AMC_fGA		AMC_dGA		AMC_hGA	
	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt
F										
01	2257.51	28710.28	2034.37	27834.65	2035.17	27760.40	**2022.40**	**28294.63**	2039.30	27768.47
02	2306.05	28672.43	2084.22	27286.12	2084.84	27671.16	**2083.97**	**27978.79**	2084.47	27944.11
03	850.24	28700.42	**608.07**	**27635.80**	608.32	27767.11	608.13	27867.36	608.63	27280.61
04	1546.96	28733.22	1329.08	26819.62	1328.77	27863.52	1329.58	28029.66	**1328.33**	**27308.04**
05	453.99	28415.05	**196.57**	**26846.99**	197.57	26783.72	197.20	28222.74	198.51	28380.85
06	855.31	28707.11	625.58	27613.17	624.96	27835.21	**624.96**	**27996.64**	625.77	27519.31
07	3980.44	28660.10	3495.86	27970.66	**3495.55**	**28138.01**	3496.73	27705.17	3496.36	28333.90

Bold text refers to the best result

40.48%, AMC_dGA 47.48% and AMC_hGA 21.43%. Closer inspection of the results is explained next.

Using a variety of operators, in which they have different performances, have provided good quality solutions for each problem set, by AMCGAs methods, especially AMC_fGA and AMC_dGA. Improvements have occurred due to the combined work of the operators, which allowed an extensive search with a larger variance than using one crossover. However, using two performance measurements, as the case in AMC_hGA was proven to be inefficient in comparison to the other AMCGAs. This is due to the large jumps in the score values in comparison to using one performance measurement, i.e. AMC_fGA and AMC_dGA, where small movements into various directions in the solution space has resulted in finding undiscovered regions easier.

By contrast, in the indirect GA, the search was completed by one crossover involvement in a large proportion of the solution space. Using one crossover throughout the search has less ability to extend the search in those regions, which were most promising. This was the reason for the GA providing worse results than the AMCGAs.

On the other hand, random crossover selection has utilised different crossovers at an arbitrary level, without prior knowledge of the current search space. Still, the AMC_rGA has obtained the best results on E_5, E_7, F_3 and F_5. This was because of the lack of performance measurements calculations. Hence, results were computed faster under those instances.

In regard to computational time, AMCGAs proved to improve the efficiency of the algorithm by computing the results in less time. For GA, AMC_rGA , AMC_fGA, AMC_dGA and AMC_hGA, the average Cpt in seconds were 6069.04, 5756.7, 5797.7, 5873.8 and 5825.6 respectively. The reason for the rapid decrease in the computational times was as follows.

In the traditional GA, one crossover was applied at each iteration. As a result, the algorithm required more time to improve the solution. On the other hand, the AMCGAs enforces the performances of different crossovers onto one population, which was made in a minimum time. Therefore, the proposed AMCGAs obtain solutions in fewer iterations than the indirect GA. In AMC_rGA , the average Cpt was the lowest among all compared methods, this was due to the exclusion of the feedback process. The average Cpt for AMC_fGA was less than the average Cpt for AMC_dGA and AMC_hGA. This result was the outcome of computing the dissimilarities between the parent and the offspring, which resulted in a massive amount of calculations. Despite the fact that in AMC_hGA the two performance measurements were applied, it obtained the results in less Cpt-time than AMC_dGA. This outcome was due to the use of the rewarding process that resulted in better and faster convergence.

5.5 Using an AMC_dGA as a component of AMCAGA

A major advantage of AMCGAs was that they have a considerable influence on improving the efficiency of the baseline GA, especially when the distance-based performance measurement was applied. Thus, the AMC_dGA method was selected to be integrated with the adaptive operator rates control method AMCAGA (noted as AMC_dAGA),

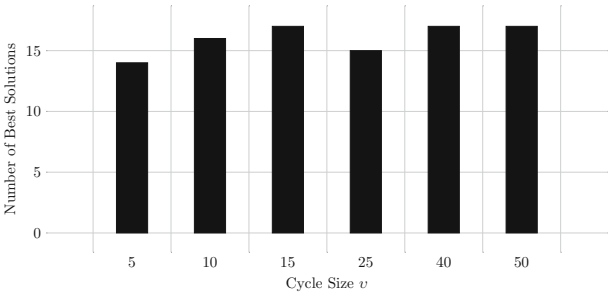

Fig. 10 Total number of best solutions while using AMC_dAGA

that perform as a full adaptive GA. The following experiments aimed to evaluate the validity of AMC_dAGA by providing further insights into its performance.

5.5.1 Effect of using different cycle sizes in AMCAGA

In this section, the aim was to investigate the effect of the using different cycle sizes in AMC_dAGA on solutions quality and computational times.

Figure 10 shows the total number of the best solutions generated by different cycle sizes. Each bar at the x-axis represents an AMC_dAGA method applied with a cycle size υ. The higher the bar the better, i.e. the more "best" values. Using different cycle sizes has resulted in relatively similar performance, with a high number of best solutions in more than one cycle size. Thus, it can be claimed that using adaptive operator rates (P_c and P_m) makes the performance of the AMCGA more stable than using the AMC_dGA separately, as seen on Figs. 6 and 10.

The number of best solutions was the highest when $\upsilon = 15$, 40 and 50 with a value of 17 best solutions. Followed by $\upsilon = 10$ with 16 solutions and $\upsilon = 25$ with 15 solutions. The values for the number of best solutions when $\upsilon = 5$ was the lowest. The time was insufficient to score all operators. Thus, a large cycle size, i.e. 50, provide better learning outcomes with the time to retrieve the information needed in order to improve the results.

Another observation was made when recording the computational times for AMC_dAGA while using different cycle sizes. Each sub-figure in Fig. 11 corresponds to a problem sets from A to F, and each bar illustrates the overall average of the computational time in seconds used by a cycle-size. The bars colour and pattern indicate each cycle as follows: black solid bars when cycle size=5, grey bars with stripped lines when $\upsilon = 10$, blue bars with dots when $\upsilon = 15$, red bars with right inclined lines when $\upsilon = 25$, green bars with left inclined lines when $\upsilon = 40$ and yellow bars with a grid when $\upsilon = 50$. The lower the bar the better, i.e. the less computational time.

On average, the lowest computational times for problem set A was when $\upsilon = 25$, however, all cycle sizes have obtained the same result in less than 50 s which was relatively acceptable. Followed by the $\upsilon = 50$ that also obtained the lowest average computational times for problem sets C, D and F. The results, as shown in Figs. 11 and 10, indicate that combining the adaptive operator rates (Pc and Pm) approach with AMC_dGA of a cycle of size 50 provides the best results in less time, especially

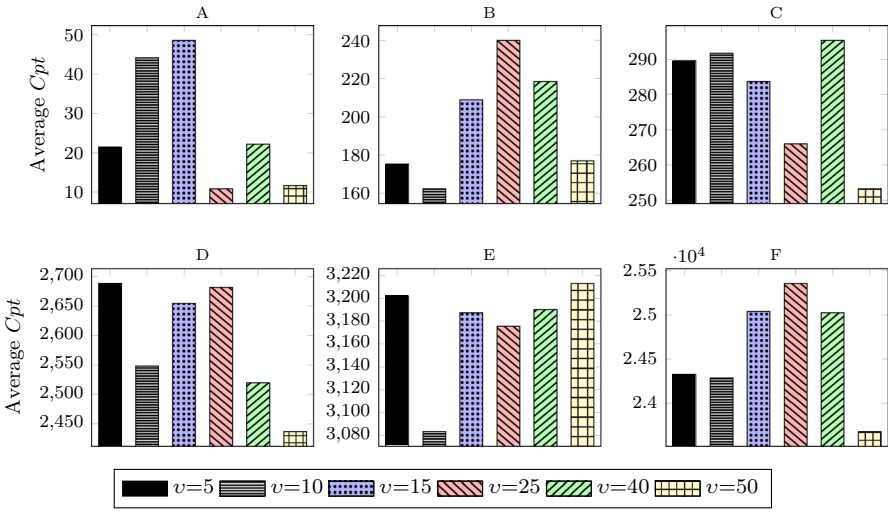

Fig. 11 Average computational-time (in seconds) produced by AMC_dAGA used with different cycle sizes υ (Colour figure online)

for difficult problem sets. The larger the cycle size, the more scores were calculated for an operator effectiveness in addition to updating the operator rates. As a result, the population evolved over time into better, fitter solutions. The interchange between the two adaptive aspects was vastly exploited in AMC_dAGA. Note that the lowest average computational times for problem sets B and E were when $\upsilon = 10$ with a difference between 100 and 200 s to the other sizes. With reference to Fig. 10, this cycle size also obtained the second highest number of best solutions. Therefore, the results for a $\upsilon = 10$ and $\upsilon = 50$ was investigated further in the next section.

5.5.2 Overall results of AMCAGA versus AMCGA and AGA

The aim of this experiments is to compare the performance of AMC_dAGA against AMC_dGA and AGA to understand the effect of combining the adaptive parameter aspects into the AMCs.

Results for AMC_dAGA are shown in Table 8 with the diversity-based adaptive operators rate control GA (noted as AGA) and the AMCGA variations that utilised diversity-based measurement (noted as AMC_dGA).

The AMC_dAGA method used in this comparison was when $\upsilon = 10$ (noted as AMC_dAGA_{10}) and a $\upsilon = 50$ (noted as AMC_dAGA_{50}). These methods are selected based on the observations taken form Figs. 6 and 7.

Note that there are two types of adaptability in Table 8. The first adaptive method used in AGA and AMC_dAGA is the GA parameter control, P_c and P_m, The second adaptive method used in AMC_dGA and AMC_dAGA is multiple crossover adaptability.

The baseline GA was excluded in this comparison because it was proven that AMC_dGA and AGA have provided better results as discussed earlier.

Table 8 Results comparison between cost values $f(S)$ and computational time Cpt (in seconds) produced by the adaptive methods AGA, AMC_dGA, AMC_dAGA_{10} and AMC_dAGA_{50} for 42 WSRP instances

	AGA		AMC_dGA		AMC_dAGA_{10}		AMC_dAGA_{50}			AGA		AMC_dGA		AMC_dAGA_{10}		AMC_dAGA_{50}	
	f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*		*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
A									**D**								
01	3.49	14.45	3.49	1.85	3.49	2.81	**3.49**	**1.78**	**01**	170.77	2978.51	170.71	3334.63	**170.23**	**2533.55**	170.29	2545.39
02	2.49	100.69	2.49	2.62	2.49	4.49	**2.49**	**1.27**	**02**	166.87	3228.52	167.31	3405.58	166.58	2109.05	**166.26**	**2512.36**
03	3.08	11.50	3.02	58.57	3.02	221.92	**3.02**	**27.63**	**03**	180.60	3258.30	180.86	3102.37	179.81	2490.46	**179.75**	**2370.22**
04	1.42	247.07	1.42	67.69	1.42	51.27	**1.42**	**42.71**	**04**	167.93	3465.12	168.37	3476.31	166.64	2382.08	**166.39**	**2511.47**
05	2.42	1.11	2.42	1.26	2.42	0.87	**2.42**	**0.38**	**05**	161.73	3367.04	161.86	3276.84	161.50	2503.26	**161.37**	**2532.9**
06	3.56	130.22	3.55	8.30	3.55	26.59	**3.55**	**7.05**	**06**	178.28	3200.96	178.16	3353.26	178.11	2765.96	**177.99**	**2368.7**
07	3.71	0.72	3.71	1.42	3.71	1.03	**3.71**	**0.51**	**07**	**178.40**	**2489.11**	178.65	3347.14	178.41	3054.21	178.41	2217.13
B									**E**								
01	1.75	84.00	1.72	276.01	**1.72**	**160.06**	1.73	146.45	**01**	1.18	3322.13	1.18	3404.99	1.17	3390.71	**1.17**	**3285.83**
02	1.75	0.56	1.75	1.22	1.75	1.03	**1.75**	**0.29**	**02**	1.21	3368.60	1.21	3399.96	**1.20**	**3303.48**	1.20	3311.01
03	1.82	280.35	1.78	570.47	**1.77**	**327.03**	1.79	468.52	**03**	1.21	3246.99	1.22	3460.71	1.21	3181.93	**1.21**	**3137.97**
04	2.08	46.19	2.08	21.25	2.08	5.12	**2.08**	**3.17**	**04**	1.29	3492.21	1.29	3493.56	**1.29**	**2932.00**	1.29	3123.67
05	1.93	98.85	1.91	483.60	1.92	459.26	**1.91**	**396.11**	**05**	2.24	3442.02	2.25	3424.19	**2.24**	**2921.37**	2.24	3378.24
06	1.66	190.74	**1.64**	**72.75**	1.65	97.23	1.64	146.05	**06**	1.30	3283.59	1.30	3356.92	**1.30**	**2539.72**	1.30	2975.27
07	1.81	154.78	1.80	244.68	1.80	87.54	**1.80**	**78.43**	**07**	1.73	3473.85	1.73	3452.58	1.72	3314.44	**1.72**	**3278.99**
C									**F**								
01	115.07	703.71	**114.75**	**566.20**	114.87	644.23	114.94	640.35	**01**	2122.60	27647.63	2022.40	28294.63	2036.82	23117.21	**2019.72**	**22709.3**
02	3.15	0.46	3.15	0.86	3.15	0.71	**3.15**	**0.30**	**02**	2082.79	26318.27	2083.97	27978.79	2081.99	23765.59	**2081.74**	**22516.9**

Table 8 continued

	AGA		AMC_dGA		AMC_dAGA_{10}		AMC_dAGA_{50}			AGA		AMC_dGA		AMC_dAGA_{10}		AMC_dAGA_{50}	
	f(S)	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*		*f(S)*	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*
03	105.08	677.45	104.89	434.64	104.75	804.69	**104.82**	**681.10**	**03**	607.26	27722.40	608.13	27867.36	605.14	22220.34	**604.71**	**22658.5**
04	11.15	2.40	**11.15**	**1.79**	11.15	3.14	11.15	2.64	**04**	1328.51	27229.31	1329.58	28029.66	1326.28	24782.11	**1325.90**	**23459**
05	12.60	33.70	12.34	68.90	12.34	2.40	**12.34**	**2.35**	**05**	196.08	26934.63	197.20	28222.74	193.59	25631.57	**192.78**	**25895.8**
06	180.99	589.30	**180.80**	**431.25**	180.92	586.95	180.99	446.39	**06**	624.77	25367.39	624.96	27996.64	622.41	26415.45	**622.34**	**23680**
07	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**07**	3495.99	27225.77	3496.73	27705.17	**3492.87**	**24083.07**	3493.18	24846.2

Bold text refers to the best result

From the table, it can be seen that AMC_dAGA has outperformed AGA and AMC_dGA, especially AMC_dAGA_{50}. Closer inspection of the results shows the percentage of best cost-values overall solutions are as follows. The AGA 21.43%, AMC_dGA 35.71%, AMC_dAGA_{10} 42.86% and AMC_dAGA_{50} 59.52%. Hence, AMC_dAGA_{50} has provided the best results on 59.52% of all solutions. Followed by AMC_dAGA_{10} that provided the best results on 42.86% of all solutions. These results indicate the power of combining two adaptive elements that work together in order to improve the GA offspring productivity.

Another advantage of this method was the rapid decrease in computational time that was previously reduced by using the adaptive methods separately in comparison to the baseline GA. On average, the computational times in seconds were 5653.11, 5873.79, 5069.67 and 4962.10 for AGA, AMC_dGA, AMC_dAGA_{10} and AMC_dAGA_{50} respectively. Thus, using the combined method was more cost-effective than using one method individually for WSRP.

5.6 Results of AMCAGA versus WSRPs solution methods

This section provides a comparison of the best-performing methods proposed in this study, i.e., AMC_dAGA_{10} and AMC_dAGA_{50}, against three existing WSRP applications: MIP solver (Laesanklang and Landa-Silva 2016), MIP with decomposition (Laesanklang et al. 2015) and VNS algorithm (Pinheiro et al. 2016). Table 9 shows the solution quality, noted as $f(S)$, and the computational time in seconds, noted as *Cpt*, in which the best solution was found. The best values are highlighted in bold.

For smaller problem sets, the proposed methods were quite competitive, matching the best-known results for many of those instances. The VNS seems to provide better overall results with 42.86% of all best solutions. However, the AMC_dAGA_{10} and AMC_dAGA_{50} outperformed the MIP with decomposition method with 21.43% and 42.86% of all best solutions respectively.

The average computational times were calculated for methods that provided solutions for all instances. i.e., MIP with decomposition AMC_dAGA_10 and AMC_dAGA_50. The recorded times were 4964.26, 5069.67 and 4962.10 respectively.

The AMC_dAGA_{10} has the largest computational time among all methods. On the contrary, the AMC_dAGA_{50} has the lowest computation time among all methods. These findings provide implications that the use of a large cycle size provided better GA performance, especially cost-wise. A noticeable effect of using adaptive aspects was the reduction of the computation time, which was also enhanced when using AMCAGA. On the other hand, the MIP with decomposition method has obtained a low average computation time. However, the results obtained were poor in comparison to the other methods.

So far, the proposed methods were the only algorithms that provided results for all instances. Hence, adaptive GAs were able to solve real-world and highly constrained optimisation problems. Interestingly, the best cost values were obtained by diversity-based methods. This further proves the significance of maintaining a diverse population in enhancing the GA performance when tackling WSRP. Even though VNS had obtained better results than the proposed methods in this study, when results were

Table 9 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by all WSRP solution methods on 42 instances

	Optimal		MIP DECOMP.		VNS		AMC_dAGA_{10}		AMC_dAGA_{50}	
	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	$Cptn$
A										
01	3.49	7.00	5.65	3.71	3.49	2.99	3.49	2.81	**3.49**	**1.78**
02	2.49	8.00	4.53	3.58	**2.49**	**0.33**	2.49	4.49	2.49	1.27
03	**3.00**	**14.00**	10.65	3.70	3.00	29.30	3.02	221.92	3.02	27.63
04	**1.42**	**5.00**	3.09	2.88	1.42	6.87	1.42	51.27	1.42	42.71
05	2.42	1.00	3.54	1.77	**2.42**	**0.11**	2.42	0.87	2.42	0.38
06	3.55	5.00	3.74	2.42	3.55	29.47	3.55	26.59	**3.55**	**7.05**
07	3.71	1.00	4.81	1.64	**3.71**	**0.07**	3.71	1.03	3.71	0.51
B										
01	**1.70**	**21.00**	1.79	8.07	1.70	156.45	1.72	160.06	1.73	146.45
02	1.75	2.00	1.89	4.29	**1.75**	**0.00**	1.75	1.03	1.75	0.29
03	1.72	6003.00	2.06	32.86	**1.72**	**473.21**	1.77	327.03	1.79	468.52
04	2.07	25.00	2.21	15.25	**2.07**	**9.03**	2.08	5.12	2.08	3.17
05	**1.82**	**585.00**	4.74	25.35	1.83	135.33	1.92	459.26	1.91	396.11
06	**1.62**	**184.00**	2.52	24.11	1.62	194.29	1.65	97.23	1.64	146.05
07	**1.79**	**300.00**	4.06	23.64	1.79	304.96	1.80	87.54	1.80	78.43
C										
01	n/a	n/a	904.98	211.64	**114.21**	**301.32**	114.87	644.23	114.94	640.35
02	3.15	6.00	3.61	0.57	**3.15**	**0.00**	3.15	0.71	3.15	0.30
03	n/a	n/a	1186.27	26.33	**103.52**	**550.29**	104.75	804.69	104.82	681.10
04	11.15	90.00	81.25	3.09	**11.15**	**0.91**	11.15	3.14	11.15	2.64
05	12.34	55.00	68.94	1.05	**12.34**	**0.34**	12.34	2.40	12.34	2.35
06	n/a	n/a	3102.26	47.05	**140.44**	**323.52**	180.92	586.95	180.99	446.39
07	4.30	1.00	5.29	0.24	4.30	0.02	**4.30**	**0.00**	**4.30**	**0.00**

Table 9 continued

	Optimal		MIP DECOMP.		VNS		AMC_dAGA_{10}		AMC_dAGA_{50}	
	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	$Cptn$
D										
01	n/a	n/a	496.4	1060.0	**170.3**	**3036.2**	170.23	2533.55	170.29	2545.39
02	n/a	n/a	372.9	1192.1	**163.9**	**2840.8**	166.58	2109.05	166.26	2512.36
03	n/a	n/a	3213.3	1209.0	**178.2**	**3172.8**	179.81	2490.46	179.75	2370.22
04	n/a	n/a	418.9	3005.1	167.1	3313.8	166.64	2382.08	**166.39**	**2511.47**
05	n/a	n/a	243.9	1306.7	**161.1**	**2818.5**	161.50	2503.26	161.37	2532.90
06	n/a	n/a	1411.3	1222.0	**177.4**	**2996.0**	178.11	2765.96	177.99	2368.70
07	n/a	n/a	753.3	1361.9	177.9	2930.4	178.41	3054.21	**178.41**	**2217.13**
E										
01	n/a	n/a	33.0	8408.0	n/a	n/a	1.17	3390.71	**1.17**	**3285.834**
02	n/a	n/a	26.0	12,448.4	n/a	n/a	**1.20**	**3303.48**	1.20	3311.008
03	n/a	n/a	29.0	20,746.6	n/a	n/a	1.21	3181.93	**1.21**	**3137.972**
04	n/a	n/a	28.5	15,190.5	n/a	n/a	**1.29**	**2932.00**	1.29	3123.67
05	n/a	n/a	270.1	32,619.2	n/a	n/a	**2.24**	**2921.37**	2.24	3378.24
06	n/a	n/a	24.6	24,212.1	n/a	n/a	**1.30**	**2539.72**	1.30	2975.266
07	n/a	n/a	427.8	51,057.3	n/a	n/a	1.72	3314.44	**1.72**	**3278.994**
F										
01	n/a	n/a	64,305.1	3446.4	n/a	n/a	2036.82	23,117.21	**2019.72**	**22,709.26**
02	n/a	n/a	73,291.2	1111.0	n/a	n/a	2081.99	23,765.59	**2081.74**	**22,516.85**
03	n/a	n/a	115,235.2	4555.1	n/a	n/a	605.14	22,220.34	**604.71**	**22,658.5**
04	n/a	n/a	102,994.2	4219.2	n/a	n/a	1326.28	24,782.11	**1325.90**	**23,458.95**
05	n/a	n/a	101,438.2	6156.5	n/a	n/a	193.59	25,631.57	**192.78**	**25,895.77**
06	n/a	n/a	76,007.1	9695.6	n/a	n/a	622.41	26,415.45	**622.34**	**23,680.04**
07	n/a	n/a	176,540.6	3832.8	n/a	n/a	**3492.87**	**24,083.07**	3493.18	24, 846.23

Bold text refers to the best result

available, this study aided in better understanding of the usability of GAs under the combined settings. This was the first time that a study on the design components of a GA, that has been used to solve WSRP.

6 Conclusion

Using synergies among the operators an provide better results than using one operator at a certain stage of the search (Li et al. 2014). This concept is used in this study by proposing an adaptive multiple crossover framework (AMCGA) to tackle the WSRP scenarios to improve the GA efficiency.

In this investigation, six different crossover operators, well-known and cost-based, are selected as part of the AMCGA method. The crossovers are rewarded with scores during the learning process, which affected their application rates during the run, and are later adjusted with values that reflect the search environment. Rewards are given to a crossover according on its effectiveness at the current stage. To evaluate a crossover, three performance measurements are utilised to provide a score for each operator. The crossover ability to provide an offspring with fitness improvements, to generate an offspring that is different than their parents or to provide fitness improvements and more diverse solutions together. Based on the performance measurements, three variations of the AMCGA are implemented (AMC_fGA, AMC_dGA, AMC_hGA) with six different cycle sizes on a total of 42 different instances of WSRP problems. The cycle-size, which provided the highest number of the best solutions, for one method is selected in a more in-depth analysis in this study.

Further analysis has identified that AMC_dGA has the best score values, the highest number of best solutions with the highest computational time among all methods. This is due to diversity-based measurement requires more computation time than fitness-based measurement. However, it can provide better results. The combination of the diversity and the fitness measurements have obtained less computational times than using separate methods. This is due to the scoring process used. If only one measurement is used; a small increase in the score occurs. On the other hand, if both performance measurements are used a large increase in the score occurs. Thus, scores in AMC_hGA are higher than AMC_fGA and AMC_dGA, henceforth, less time is required.

Another key finding is that the highest average application rate among all methods is obtained by FCX and PMFCX crossovers. Results showed that PMFCX crossover is the most utilised crossover in all methods.

Results are compared with MIP with decomposition (Laesanklang et al. 2015), VNS algorithm (Pinheiro et al. 2016), indirect GA (non-adaptive) (Algethami et al. 2016), randomly uniform variation of the $AMCGA$ (no-learning), noted as AMC_rGA and adaptive parameter control GA (AGA) (Algethami and Landa-Silva 2017). In general, the AMCGAs have outperformed the stated methods, especially when diversity measurement is applied. Using a learning scheme has effectively improved the GA performance.

The key strength of this study is the combinations of adaptive GA settings (P_c and P_m rates) in addition to the adaptive multiple crossover method (AMCGA). This com-

bination has resulted in better quality solutions, in less computational times. Hence, using a full adaptive algorithm is the best variation of a GA to be used for the combined scheduling and routing problem settings. Further work needs to be carried out to validate the features of the set of operators used in this study, such as the number of operators and the type of operators.

References

Algethami, H., Landa-Silva, D., Martinez-Gavara, A.: Selecting genetic operators to maximise preference satisfaction in workforce scheduling and routing problem. In: Proceedings of the 6th International Conference on Operations Research and Enterprise Systems (ICORES), Porto, Portugal, pp. 416–423 (2017)

Algethami, H., Landa-Silva, D.: Diversity-based adaptive genetic algorithm for a workforce scheduling and routing problem. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1771–1778 (2017)

Algethami, H., Pinheiro, R.L., Landa-Silva, D.: A genetic algorithm for a workforce scheduling and routing problem. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 927–934 (2016)

Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2), 235–256 (2002)

Belluz, J., Gaudesi, M., Squillero, G., Tonda, A.: Operator selection using improved dynamic multi-armed bandit. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, pp. 1311–1317. ACM, New York (2015)

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints. In: International Conference on Operations Research and Enterprise Systems (ICORES), pp. 367–375. Scipress, Lisbon, Portugal (2015)

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: Workforce scheduling and routing problems: literature survey and computational study. Ann. Oper. Res. **239**(1), 39–67 (2016)

Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. J. Heuristics **42**(1), 63–86 (1998)

Consoli, P., Yao, X.: Diversity-driven selection of multiple crossover operators for the capacitated Arc routing problem. In: Blum, C., Ochoa, G. (eds.) Evolutionary Computation in Combinatorial Optimisation: 14th European Conference, EvoCOP 2014, Granada, Spain, April 23–25, Revised Selected Papers, pp. 97–108. Springer, Berlin (2014)

Contreras-Bolton, C., Gatica, G., Barra, C.R., Parada, V.: A multi-operator genetic algorithm for the generalized minimum spanning tree problem. Expert Syst. Appl. **50**, 1–8 (2016)

Cowling, P., Colledge, N., Dahal, K., Remde, S.: The trade-off between diversity and quality for multi-objective workforce scheduling. In: Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP'06, pp. 13–24. Springer, Berlin (2006)

Črepinšek, M., Liu, S.-H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM Comput. Surv. **45**(3), 35:1–35:33 (2013)

Eiben, A.E., Horvath, M., Kowalczyk, W., Schut, M.C.: Reinforcement learning for online control of evolutionary algorithms. In: Engineering Self-Organising Systems: 4th International Workshop, ESOA 2006, Hakodate, Japan, May 9, 2006, Revised and Invited Papers, pp. 151–160. Springer, Berlin (2007)

Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Ann. Math. Artif. Intell. **60**(1), 25–64 (2010)

García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J. Heuristics **15**(6), 617 (2008)

Hartmann, S.: A competitive genetic algorithm for resource-constrained project scheduling. Naval Res. Logist. NRL **45**(7), 733–750 (1998)

Laesanklang, W., Landa-Silva, D.: Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. Ann. Oper. Res. **169**, 1–35 (2016)

Laesanklang, W., Lankaites-Pinheiro, R., Algethami, H., Landa-Silva, D.: Extended decomposition for mixed integer programming to solve a workforce scheduling and routing problem. In: de Werra,

D., Parlier, G.H., Vitoriano, B. (eds.) Operations Research and Enterprise Systems: 4th International Conference, ICORES 2015: Revised Selected Papers, Volume 577 of Communications in Computer and Information Science, pp. 191–211. Springer, Lisbon (2015)

Lassaigne, R., De Rougemont, M.: Logic and Complexity. Springer, New York (2012)

Li, K., Fialho, A., Kwong, S., Zhang, Q.: Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **18**(1), 114–130 (2014)

Mankowska, D., Meisel, F., Bierwirth, C.: The home health care routing and scheduling problem with interdependent services. Health Care Manag. Sci. **17**(1), 15–30 (2014)

Maturana, J., Saubion, F.: A compass to guide genetic algorithms. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) Proceedings of the Parallel Problem Solving from Nature—PPSN X: 10th International Conference, Dortmund, Germany, September 13–17, 2008, pp. 256–265. Springer, Berlin (2008)

Misir, M., Smet, P., Verbeeck, K., Vanden Berghe, G.: Security personnel routing and rostering: a hyper-heuristic approach. In: Proceedings of the 3rd International Conference on Applied Operational Research, vol. 3, Tadbir, pp. 193–205 (2011)

Morrison, R.W., De Jong, K.A.: Measurement of population diversity. In: Artificial Evolution: 5th International Conference, Evolution Artificielle, EA 2001 Le Creusot, France, October 29–31, 2001 Selected Papers, pp. 31–41. Springer, Berlin (2002)

Mutingi, M., Mbohwa, C.: Health-care staff scheduling in a fuzzy environment: a fuzzy genetic algorithm approach. In: Conference Proceedings (DFC Quality and Operations Management), International Conference on Industrial Engineering and Operations Management, pp. 303–312 (2014)

Onieva, E., Osaba, E., Angulo, I., Moreno, A., Bahillo, A., Perallos, A.: Improvement of drug delivery routes through the adoption of multi-operator evolutionary algorithms and intelligent vans capable of reporting real-time incidents. IEEE Trans. Autom. Sci. Eng. **14**(2), 1009–1019 (2017)

Osaba, E., Onieva, E., Carballedo, R., Diaz, F., Perallos, A.: An adaptive multi-crossover population algorithm for solving routing problems. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), pp. 113–124. Springer (2014)

Osaba, E., Onieva, E., Carballedo, R., Diaz, F., Perallos, A., Zhang, X.: A multi-crossover and adaptive island based population algorithm for solving routing problems. J. Zhejiang Univ. Sci. C **14**(11), 815–821 (2013)

Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. Auton. Agents Multi-agent Syst. **11**(3), 387–434 (2005)

Pinheiro, R.L., Landa-Silva, D., Atkin, J.: A variable neighbourhood search for the workforce scheduling and routing problem. In: Advances in Nature and Biologically Inspired Computing, pp. 247–259 . Springer, Pietermaritzburg, South Africa (2016)

Puljić, K., Manger, R.: Comparison of eight evolutionary crossover operators for the vehicle routing problem. Math. Commun. **18**(2), 359–375 (2013)

Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J.: The home care crew scheduling problem: preference-based visit clustering and temporal dependencies. Eur. J. Oper. Res. **219**(3), 598–610 (2012)

Spears, W.M.: Adapting crossover in evolutionary algorithms. In: Proceedings of the Fourth Annual Conference on Evolutionary Programming, pp. 367–384. MIT Press (1995)

Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO '05, pp. 1539–1546. ACM, New York (2005)

Tuson, A.L.: Adapting operator probabilities in genetic algorithms. Technical report, Master's thesis, Evolutionary Computation Group, Department of Artificial Intelligence, Edinburgh University (1995)

Whitacre, J.M., Pham, T.Q., Sarker, R.A.: Use of statistical outlier detection method in adaptive evolutionary algorithms. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, pp. 1345–1352. ACM, New York (2006)